NASA CR-177,963

NASA Contractor Report 177963

NASA-CR-177963
19860007436

CARE III, Version 4 Enhancements

L.A. Bryant and J.J. Stiffler

Sequoia Systems, Inc.
Marlborough, Massachusetts 01752

Contract NAS1-17476
November 1985

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

1 Report No	2 Government Accession No	3 Recipient's Catalog No
NASA CR-177963		

4 Title and Subtitle	5 Report Date
CARE III, Version 4 Enhancements	November 1985
	6 Performing Organization Code

7 Author(s)	8 Performing Organization Report No
L. A. Bryant and J. J. Stiffler	
	10 Work Unit No
9 Performing Organization Name and Address	
Sequoia Systems, Inc. Marlborough, MA 01752	11 Contract or Grant No NAS1-17476
	13 Type of Report and Period Covered Contractor Report
12 Sponsoring Agency Name and Address	
National Aeronautics and Space Administration Washington, DC 20546	14 Sponsoring Agency Code 505-34-13-30

15 Supplementary Notes
NASA Project Engineer: Salvatore J. Bavuso
NASA Langley Research Center
Hampton, VA 23665

16 Abstract

This report delineates the enhancements and error corrections to CARE III Version 4. All changes to Version 4 with the exception of the internal redundancy model have been implemented in Version 5. Version 4 is the first public release version for execution on the CDC Cyber 170 series computers. Version 5 is the second release version and it is written in ANSI standard Fortran 77 for execution on the DEC VAX 11/700 series computers and many others.

17 Key Words (Suggested by Author(s))	18 Distribution Statement
Reliability modeling CARE III Fault coverage Fault models Fault-tolerant	Unclassified - Unlimited Subject Category 59

19 Security Classif (of this report)	20 Security Classif (of this page)	21 No of Pages	22 Price
Unclassified	Unclassified	141	A07

N-305 For sale by the National Technical Information Service Springfield Virginia 22161

TABLE OF CONTENTS

1.0 Introduction..... 1

2.0 CARE III Conversion to FORTRAN 77..... 3

2.1 Code Conversion Changes..... 3

2.2 Code Conversion Testing..... 9

3.0 Examination and Enhancement of Stiff Volterra Integral Equation Solution..... 13

3.1 Gear's Method..... 13

3.2 Smotherman (Picard) Iteration..... 14

3.3 Addition of Adaptive Step Size Halving Capability.... 14

4.0 Examination and Enhancement of Kolmogorov Forward Equation Solution..... 19

4.1 Software Design Changes..... 19

4.1.1 Review of Original Software Design..... 20

4.1.2 Enhanced Software Design..... 22

4.1.2.1 Computing the Doubling Parameters..... 22

4.1.2.2 Determining a Valid Set of CVGSTP's..... 24

4.1.2.3 Selecting the Most Valid CVGSTP..... 26

4.1.3 Results of Enhanced Software Design..... 26

4.2 Code Changes..... 28

4.3 Testing of the Code Changes..... 30

4.4 Enhancement Results..... 33

5.0 Elimination of Oscillation in Module CARE3 Functions..... 38

5.1 Code Changes Related to Function Oscillation Problems 38

5.2 Testing and Results of Code Changes..... 40

TABLE OF CONTENTS (CONCLUDED)

6.0 Consolidation of Boeing's Version 4 and Enhanced CARE III Code..... 48

6.1 Code Conversion and Insertion..... 48

6.2 Testing of Consolidated Code..... 54

6.3 Code Errors Found During COVRGE Consolidation..... 55

6.4 Code Errors Found During CARE3 Consolidation..... 57

7.0 Implementation of Internally Redundant Modeling Capability 63

7.1 Mathematical Model..... 64

7.1.1 Derivation of Mathematical Model..... 68

7.1.2 Limitation of Mathematical Model..... 72

7.2 Added Input Parameters..... 72

7.3 Code Additions and Modifications..... 75

7.4 Testing of the Enhancement..... 77

8.0 Description of Major CARE III Data Structures..... 87

8.1 Coverage Plotting Files Data Structure..... 87

8.2 Reliability Plotting File Data Structure..... 91

8.3 COMMON Block BXYCOM Data Structure..... 92

9.0 Additional Testing of User Input Values..... 107

10.0 Concluding Remarks..... 111

11.0 References..... 113

Appendix A - Symbolic Name Equates..... 115

Appendix B - Example Problem 5 Output Listings..... 119

Appendix C - FTMP Output Listing..... 133

LIST OF FIGURES

Figure 2-1 - NASA NAMELIST Input File CASE1.DAT..... 9

Figure 2-2 - NASA Generated Output from CASE1.DAT..... 10

Figure 2-3 - List Directed Format Input File CASE1.DAT..... 10

Figure 2-4 - FORTRAN 77 CASE1.DAT Output Results..... 10

Figure 2-5 - NASA NAMELIST Input File CASE2.DAT..... 11

Figure 2-6 - List Directed Format Input File CASE2.DAT..... 12

Figure 2-7 - FORTRAN 77 CASE2.DAT Output Results..... 12

Figure 3-1 - List Directed Input File Test Case 3d..... 16

Figure 3-2 - List Directed Input File Test Case 3d''..... 16

Figure 5-1 - Graph of Failure Rates for Test Cases 5, 6 and 7..... 47

Figure 7-1 - FTMP Original Test Case (pre-internally redundant modeling capability format)..... 80

Figure 7-2 - FTMP Modified Test Case (without internally redundant modules)..... 81

Figure 7-3 - FTMP Modified Test Case (with internally redundant memory modules)..... 82

Figure 9-1 - FTMP Modified Test Case (containing input errors)..... 108

Figure 9-2 - CAREIN Output Listing (generated from error filled input file)..... 110

LIST OF TABLES

Table 3-1 - Comparison of Final Result Using Various Versions of CARE III..... 17

Table 4-1 - Number of Unique Coverage Values Passed to CARE3 as FT Increases (Using Equal Step Sizes)... 21

Table 4-2 - (NPERST, NDUB, NSTEPS) Vector Solutions Independent of FT..... 25

Table 4-3 - Increased Number of Unique Coverage Values with Enhanced COVRGE..... 27

Table 4-4 - Example Problem 5 Results at 10 hours..... 34

Table 4-5 - Example Problem 5 Results at 1600 hours..... 35

Table 4-6 - Example Problem 5 Results at 80000 hours..... 35

Table 5-1 - Elimination of Function Oscillation Test Cases... 41

Table 5-2 - Permanent Test Case Results at 60 min..... 46

Table 7-1 - Failure Rates for FTMP Test Cases..... 86

Table 7-2 - FTMP Test Case Results at 100 min..... 86

Table 8-1 - Description of BXYCOM Variables..... 95

Table 8-2 - N-XY Critical Pair Counts Per Subrun..... 96

Table 8-3 - (ISTG) Single-Fault Reliability Functions Per Subrun..... 97

Table 8-4 - Portion of BXYAR Generated in CAREIN Module Per Subrun..... 98

Table 8-5 - (ISTG,ISTG) Double-Fault Reliability Functions Per Subrun..... 99

Table 8-6 - (ISTG,JSTG) Double-Fault Reliability Functions Per Subrun..... 100

Table 8-7 - Number of Stage Pairs Given Number of Critically Coupled Stages..... 103

1.0 INTRODUCTION

CARE III (Computer-Aided Reliability Estimation, version 4) is a computer program that predicts the unreliability of highly reliable reconfigurable fault-tolerant systems. Its predecessor CARE III, version 3, developed at Raytheon Co. under the direction of Dr. J. J. Stiffler, was delivered to the NASA Langley Research Center in 1982. Care III, version 3, was reviewed and modified by Boeing Computer Services under the direction of D. M. Rose yielding version 4 (refs. 5 and 6).

Under NASA funding and direction Sequoia Systems, Inc. was to make enhancements to the mathematical model and computer program, version 4. The following are the enhancements made to CARE III and that are detailed in this report:

1. CARE III conversion to FORTRAN 77;
2. Examination and enhancement of the stiff Volterra integral equation solution;
3. Examination and enhancement of the Kolmogorov forward equation solution;
4. Elimination of the oscillation in the module CARE3 functions;
5. Consolidation of Boeing's version 4 and the enhanced CARE III code;
6. Implementation of internally redundant modeling capability;
7. Description of the major CARE III data structures;
8. Additional testing of user input values.

While implementing the enhancements listed above, several coding errors were found and corrected. Some were found in the original CARE III, version 3 code; some were found in version 4 due to the conversion from the CDC* 60 bit version to the VAX** 32 bit version; and some were found in the modifications and additions made by Boeing Computer Services to yield version 4. These errors and corrections are throughly detailed in this report.

---

* CDC is a Trademark of Control Data Corporation.

** VAX is a Trademark of Digital Equipment Corporation.

2.0 CARE III CONVERSION TO FORTRAN 77

The original CARE III version 4, VAX implementation code, was successfully compiled and loaded using Bell Laboratories' "Portable FORTRAN 77 Compiler", implemented on a VAX 11/750 under the Berkeley 4.1 UNIX* operating system. There were five types of necessary changes made to the routines in order to make them FORTRAN 77 standard, and thus enable them to compile using this system. The five problem areas, which follow, will be addressed below:

1) Seven character symbolic names,
2) Illegal comparison of a non-character variable to a character string,
3) Impossible conversion of a character string to a non-character variable,
4) NAMELIST input, and
5) Use of I/O units greater than 19.

2.1 CODE CONVERSION CHANGES

All seven character symbolic names were changed to six characters in CAREIN, COVRGE and CARE3. The M4 macro processor was used to implement the name changes. Most variables could have been truncated to use the first six characters, but in most cases this would have yielded an even less meaningful name. Therefore, vowels were removed as much as possible. (See appendix A for all symbolic name equates including those converted in plotting programs CVGPLT and RELPLT.)

---

* UNIX is a Trademark of Bell Laboratories.

In all three main programs, non-character variable TBASE was compared to Hollerith constants. This comparison was changed to use an array containing the equivalent character strings.

Example:

```
IF(TBASE.EQ.4HHRS) FTHRS = FT    was changed to:

IF(TBASE.EQ.TBSAR(1)) FTHRS = FT    where TBSAR was defined as:

DIMENSION TBSAR(4)
DATA TBSAR/'HRS ','MINS','SECS','MSEC'/
```

In CARE3, real variable PRCODE was compared to Hollerith constants. PRCODE was declared CHARACTER*4 in all subroutines containing COMMON /STEPCM/.

In COVRGE, DOUBLE PRECISION debug variable ARNME was set to a HOLLERITH constant. It was changed to have a CHARACTER*8 declaration in all subroutines.

In CAREIN, the FORTRAN 77 non-standard NAMELIST input method was replaced with list directed formatting. List directed formatting was chosen because it is very similar to the NAMELIST method. Both methods allow for defaults to be used if a variable is not given a value in the input stream, and both use free format input. The NAMELIST method requires that the data variable names be specified and equated to their desired values; therefore, order is not important. List directed formatting is order dependent because the data variable name is not specified in the input stream. To use a default for a data variable, an extra comma is required instead of a value.

Example:

NAMELIST input:

```
$FLTTYP  NFTYPS=2, ALP=2*100.0, BET=2*1000.0, DEL=100.0,360.0, EPS=2*0.0,
         IDELF = 2*1, IEPSF = 2*1, PB=2*1.0, TRUNC=1.0E-5$
```

List directed formatting input:

```
 2, 100.0, 100.0, 1000.0, 1000.0, 100.0, 360.0,,, 0.0, 0.0, 1, 1,,,
 1, 1,,, 1.0, 1.0,,,, 1.0E-5/
```

Unfortunately, this method is more error prone due to the necessary placement of the extra commas. But with the use of the menu input preprocessor CARE3MENU (ref. 1), this would not be a problem. Unfortunately, the current version of CARE3MENU runs solely under the VAX computers' VMS operating system and not under UNIX where list directed formatting is used. CARE3MENU would have to be changed to run under UNIX and to generate this new style of input, if a completely FORTRAN 77 standard CARE III program is desired which includes the input preprocessor.

The pertinent CAREIN code change follows:

```
C
C  USE LIST DIRECTED FORMATTING INSTEAD OF NAMELISTS.
C
C  $FLTTYP INPUT
      READ(7,*,END=10) NFTYPS, (ALP   (ITYP),ITYP=1,NFTYPS),
     .                         (BET   (ITYP),ITYP=1,NFTYPS),
     .                         (DEL   (ITYP),ITYP=1,NFTYPS),
     .                         (RHO   (ITYP),ITYP=1,NFTYPS),
     .                         (EPS   (ITYP),ITYP=1,NFTYPS),
     .                         (IDELF(ITYP),ITYP=1,NFTYPS),
     .                         (IRHOF(ITYP),ITYP=1,NFTYPS),
     .                         (IEPSF(ITYP),ITYP=1,NFTYPS),
     .                         (PA    (ITYP),ITYP=1,NFTYPS),
     .                         (PB    (ITYP),ITYP=1,NFTYPS),
     .                         (C     (ITYP),ITYP=1,NFTYPS),
     .                        DBLDF, TRUNC, CVPRNT, CVPLOT,
     .                        IAXSCV, MARKOV, LGTMST
C
```

(CAREIN code change continued)

```
C  $STAGES INPUT
      READ(7,*,END=10) NSTGES, (N    (ISTG),ISTG=1,NSTGES),
     .                          (M    (ISTG),ISTG=1,NSTGES),
     .                          (NSUB(ISTG),ISTG=1,NSTGES),
     .                          (MSUB(ISTG),ISTG=1,NSTGES),
     .                          (ACSP(ISTG),ISTG=1,NSTGES),
     .                         ((NOP  (IQ,ISTG),IQ=1,5),ISTG=1,NSTGES),
     .                          (LC   (ISTG),ISTG=1,NSTGES),
     .                          IRLPCD, RLPLOT, IAXSRL
C
C  $FLTCAT INPUT
      READ(7,*,END=10)  (NFCATS(ISTG),ISTG=1,NSTGES),
     .                 ((JTYP   (ICAT,ISTG),ICAT=1,NFCATS(ISTG))
     .                                      ,ISTG=1,NSTGES),
     .                 ((JSBTYP(ICAT,ISTG),ICAT=1,NFCATS(ISTG))
     .                                      ,ISTG=1,NSTGES),
     .                 ((OMG    (ICAT,ISTG),ICAT=1,NFCATS(ISTG))
     .                                      ,ISTG=1,NSTGES),
     .                 ((RLM    (ICAT,ISTG),ICAT=1,NFCATS(ISTG))
     .                                      ,ISTG=1,NSTGES),
     .                 ((OMGSUB(ICAT,ISTG),ICAT=1,NFCATS(ISTG))
     .                                      ,ISTG=1,NSTGES),
     .                 ((RLMSUB(ICAT,ISTG),ICAT=1,NFCATS(ISTG))
     .                                      ,ISTG=1,NSTGES)
C
C  $RNTIME INPUT
      READ(7,*,END=10) FT, NSTEPS, ITBASE, SYSFLG, CPLFLG, KWT, PSTRNC,
     .                 QPTRNC, NPSBRN, CINDBG, CKDATA
C
```

The four READ statements listed above contain all input parameters for the enhanced version of CARE III. (See section 7.2 and reference 2 for a detailed description of all input parameters.) The order of the variables in the READ statements is the order that the variables must be specified when using the list directed input format.

In CAREIN, I/O units 25 (file FT25F) and 26 (file FT26F) were changed to use units 17 and 18 - file names were left unchanged.

This change was necessary because this system allows only use of units 0 - 19. (Note that in the final enhanced version of CAREIN, file FT26F is no longer used.)

In order to eliminate I/O problems, OPEN and REWIND statements were inserted into each of the main programs. The following lists are comprised of those statements contained in the final enhanced version of CARE III:

CAREIN:

```
OPEN(UNIT=3,FILE='COVIN',FORM='UNFORMATTED',STATUS='NEW')
OPEN(UNIT=4,FILE='RELIN',FORM='UNFORMATTED',STATUS='NEW')
OPEN(UNIT=7,FILE='CREIN',STATUS='OLD')
OPEN(UNIT=8,FILE='CREOUT',STATUS='NEW')
OPEN(UNIT=10,FILE='FT10F',STATUS='NEW')
OPEN(UNIT=14,FILE='BXYFL',FORM='UNFORMATTED',STATUS='NEW')
OPEN(UNIT=15,FILE='FT15F',FORM='UNFORMATTED',STATUS='NEW')
OPEN(UNIT=17,FILE='FT25F',STATUS='NEW')
REWIND 3
REWIND 4
REWIND 7
REWIND 8
REWIND 10
REWIND 14
REWIND 15
REWIND 17
```

COVRGE:

```
OPEN(UNIT=3,FILE='COVIN',FORM='UNFORMATTED',STATUS='OLD')
OPEN(UNIT=7,FILE='CVGMTS',FORM='UNFORMATTED',STATUS='NEW')
OPEN(UNIT=8,FILE='DBUG',STATUS='NEW')
OPEN(UNIT=9,FILE='SNGFL',FORM='UNFORMATTED',STATUS='NEW')
OPEN(UNIT=10,FILE='DBLFL',FORM='UNFORMATTED',STATUS='NEW')
REWIND 3
REWIND 7
REWIND 8
REWIND 9
REWIND 10
```

CVGPLT:

```
OPEN(UNIT= 3,FILE='COVIN',FORM='UNFORMATTED',STATUS='OLD')
OPEN(UNIT= 4,FILE='PLFILE',STATUS='UNKNOWN')
OPEN(UNIT= 9,FILE='SNGFL',FORM='UNFORMATTED',STATUS='OLD')
OPEN(UNIT=10,FILE='DBLFL',FORM='UNFORMATTED',STATUS='OLD')
REWIND 3
REWIND 4
REWIND 9
REWIND 10
```

CARE3:

```
OPEN(UNIT=4,FILE='RELIN',FORM='UNFORMATTED',STATUS='OLD')
OPEN(UNIT=7,FILE='CVGMTS',FORM='UNFORMATTED',STATUS='OLD')
OPEN(UNIT=8,FILE='DBUG',STATUS='UNKNOWN')
OPEN(UNIT=13,FILE='PRFNCS',STATUS='NEW')
OPEN(UNIT=14,FILE='BXYFL',FORM='UNFORMATTED',STATUS='UNKNOWN')
OPEN(UNIT=15,FILE='FT15F',FORM='UNFORMATTED',STATUS='OLD')
OPEN(UNIT=16,FILE='PLTFL',FORM='UNFORMATTED',STATUS='NEW')
OPEN(UNIT=17,FILE='SCR17',FORM='UNFORMATTED',STATUS='NEW')
OPEN(UNIT=18,FILE='INXY',FORM='UNFORMATTED',STATUS='NEW')
OPEN(UNIT=19,FILE='IBXY',FORM='UNFORMATTED',STATUS='NEW')
REWIND 4
REWIND 7
REWIND 8
REWIND 13
REWIND 14
REWIND 15
REWIND 16
REWIND 17
REWIND 18
REWIND 19
```

RELPLT:

```
OPEN(UNIT= 2,FILE='RELIN',FORM='UNFORMATTED',STATUS='OLD')
OPEN(UNIT= 4,FILE='PLFILE',STATUS='UNKNOWN')
OPEN(UNIT=16,FILE='PLTFL',FORM='UNFORMATTED',STATUS='OLD')
REWIND 2
REWIND 4
REWIND 16
```

These file definitions correspond to the file definitions, on the PROGRAM statements, in the CDC version of CARE III.

2.2 CODE CONVERSION TESTING

The original CARE III version 4, FORTRAN 77 programs were successfully executed on a VAX 11/750. The two test cases sent by NASA, with the original code, were rewritten using the list directed formatting input method, described in section 2.1. Figures 2-1 and 2-5 contain the two NASA input files along with the final portion of the corresponding output file for the first test case (figure 2-2). Figures 2-3 and 2-6 contain the two files executed on the VAX 11/750 along with the final portion of their corresponding output files (figures 2-4 and 2-7).

```
 $FLTTYP  DEL(  1)  = 1.0E+1,
          DBLDF       = 0.01,
          CVPRNT      = .TRUE.$
 $STAGES  NSTGES      = 2,
          N(  1)  = 2,
          M(  1)  = 1,
          N(  2)  = 2,
          M(  2)  = 1,
          IRLPCD      = 4$
 $FLTCAT  NFCATS(  1)  = 1,
          JTYP(1,1)  = 1,
          OMG(1,1)  = 1.0,
          RLM(1,1)  = 1.0E-1,
          NFCATS(  2)  = 1,
          JTYP(1,2)  = 1,
          OMG(1,2)  = 1.0,
          RLM(1,2)  = 1.0E-1$
 $RNTIME  FT          = 1.0,
          SYSFLG      = .TRUE.,
          CPLFLG      = .TRUE.,
          CINDBG      = .TRUE.$
CASER2A : TO TEST SYSTEM FAULT TREE - 'AND' TREE
 1 2 3 3                  .
 3 A 1 2
CRITICAL PAIR TREE - ALL PAIRS ARE CRITICAL
 1 4 5 5
 1 1 2
 2 3 4
 5 2 1 2 3 4
```

Figure 2-1 - NASA NAMELIST Input File CASE1.DAT

```
NUMBER OF FAILED        UNRELIABILITY AT        PERFECT COVERAGE
                                                 UNRELIABILITY
     STAGES                 1.0000  HRS        AT    1.0000  HRS

       0                3.1488777604E-03        0.0000000000E+00
       1                2.0517311059E-03        0.0000000000E+00
       2                X                       8.2009588368E-05

TOTAL SYSTEM UNRELIABILITY AT    1.0000  HRS = 5.2826185711E-03
```

Figure 2-2 - NASA Generated Output from CASE1.DAT

```
,,, 10.0,,,,,,,,, 0.01,, .TRUE./
2, 2, 2, 1, 1,,,,,,,,,,,,, 4/
1, 1, 1, 1,,, 0.1, 0.1
1.0 ,,, .TRUE. , .TRUE. ,,, .TRUE./
CASER2A : TO TEST SYSTEM FAULT TREE - 'AND' TREE
 1 2 3 3
 3 A 1 2
CRITICAL PAIR TREE - ALL PAIRS ARE CRITICAL
 1 4 5 5
 1 1 2
 2 3 4
 5 2 1 2 3 4
```

Figure 2-3 - List Directed Format Input File CASE1.DAT

```
NUMBER OF FAILED        UNRELIABILITY AT        PERFECT COVERAGE
                                                 UNRELIABILITY
     STAGES                 1.0000  HRS        AT    1.0000  HRS

       0                3.1488784589e-03        0.0000000000e+00
       1                2.0517315716e-03        0.0000000000e+00
       2                X                       8.2009588368e-05

TOTAL SYSTEM UNRELIABILITY AT    1.0000  HRS = 5.2826195024e-03
```

Figure 2-4 - FORTRAN 77 CASE1.DAT Output Results

```
 $FLTTYP  NFTYPS       = 1     ,
          ALP(  1)  = 1.0E+1,
          BET(  1)  = 1.0E+1,
          DEL(  1)  = 1.0E+1,
          RHO(  1)  = 1.0E+1,
          EPS(  1)  = 2.0E+1,
          IDELF(  1)  = 1     ,
          IRHOF(  1)  = 1     ,
          IEPSF(  1)  = 1     ,
          PA(  1)  = 1.0   ,
          PB(  1)  = 1.0   ,
          C(  1)  = 0.5   ,
          DBLDF        = 0.01  ,
          CVPRNT       = .TRUE.$
 $STAGES  NSTGES       = 2     ,
          N(  1)  = 2     ,
          M(  1)  = 1     ,
          N(  2)  = 2     ,
          M(  2)  = 1     ,
          IRLPCD       = 4     $
 $FLTCAT  NFCATS(  1)  = 1     ,
          JTYP(1,1)  = 1     ,
          OMG(1,1)  = 1.0   ,
          RLM(1,1)  = 1.0E-1,
          NFCATS(  2)  = 1     ,
          JTYP(1,2)  = 1     ,
          OMG(1,2)  = 1.0   ,
          RLM(1,2)  = 1.0E-1$
 $RNTIME  FT           = 1.0   ,
          SYSFLG       = .TRUE.,
          CPLFLG       = .TRUE.,
          CINDBG       = .TRUE.$
CASE2 : TO TEST SYSTEM FAULT TREE - 'AND' TREE
 1 2 3 3
 3 A 1 2
CRITICAL PAIR TREE - ALL PAIRS CRITICAL
 1 4 5 5
 1 1 2
 2 3 4
 5 2 1 2 3 4
```

Figure 2-5 - NASA NAMELIST Input File CASE2.DAT

```
1, 4*10.0, 20.0, 3*1, 2*1.0, 0.5, 0.01,, .TRUE./
3*2, 2*1,,,,,,,,,,,,,4/
4*1, 2*1.0, 2*1.0E-1
1.0,,, .TRUE., .TRUE.,,, .TRUE./
CASE2 : TO TEST SYSTEM FAULT TREE - 'AND' TREE
 1 2 3 3
 3 A 1 2
CRITICAL PAIR TREE - ALL PAIRS CRITICAL
 1 4 5 5
 1 1 2
 2 3 4
 5 2 1 2 3 4
```

Figure 2-6 - List Directed Format Input File CASE2.DAT

```
NUMBER OF FAILED        UNRELIABILITY AT        PERFECT COVERAGE
                                                 UNRELIABLITY
     STAGES                  1.0000  HRS        AT    1.0000  HRS

       0                 8.274398744le-02       0.0000000000e+00
       1                 7.7691860497e-03       0.0000000000e+00
       2                 X                      8.2009588368e-05

TOTAL SYSTEM UNRELIABILITY AT    1.0000  HRS = 9.0595178306e-02
```

Figure 2-7 - FORTRAN 77 CASE2.DAT Output Results

The two CASE1.DAT results agree to six decimal places. Sequoia's VAX 11/750 uses software floating point calculations, which may account for the slight discrepancy in the two results. The second test case sent by NASA, CASE2.DAT, did not have a corresponding output file. Therefore a comparison of the two results is not possible.

3.0 EXAMINATION AND ENHANCEMENT OF STIFF VOLTERRA INTEGRAL EQUATION SOLUTION

Stiff integration methods were examined for solving the coverage Volterra integral equations. None were found applicable to the particular type of functions contained in the CARE III coverage model (refs. 5 and 8).

An enhancement was made to the existing COVRGE integration method to reduce its stability problems mentioned in reference 9. This enhancement is detailed in section 3.3.

3.1 GEAR'S METHOD

A good deal of research was done into applying Gear's method to the COVRGE Volterra equations. It was determined that solving the Volterra equations, contained in module COVRGE, using Gear's method is basically impossible. The COVRGE Volterra equations cannot be converted to ordinary differential equations, which Gear's method requires (refs. 3 and 4), because the kernels of the Volterra equations used in CARE III do not contain a finite number of non-zero derivatives.

3.2 SMOTHERMAN (PICARD) ITERATION

Smotherman applied this method to a transient fault model (ref. 7). CARE III's method of handling transients is different:

- In Smotherman's Dissertation Defense, the series converged since the ith term was bounded by $(\lambda t)^i$, with λ = transient failure rate and t = mission duration, so that λt was typically very small.
- In the CARE III coverage model, however, the ith term can be of the order of αt with α the active-to-benign transition rate and t the fault recovery time. This αt is not necessarily small, and the series convergent rate may be very slow indeed.

It is not possible to apply the Smotherman (Picard) technique to the coverage model Volterra equations due to the slow series convergent rate.

3.3 ADDITION OF ADAPTIVE STEP SIZE HALVING CAPABILITY

In the CARE III Phase III report (ref. 9), J. J. Stiffler recommended that the coverage functions' adaptive step size be allowed to halve as well as double. This recommendation was made due to the fact that the coverage functions do not have monotonically decreasing derivatives. This modification was made to the coverage program.

This modification basically entailed a change in the way that the step count per step size change was stored per function calcula-

tion. For example: (NSTPAR(I),I=1,64) = 2 1 1 1 80 0 ... 0 tells us that there were two steps using the initial step size, one step for the next three doublings of the step size and 80 steps using the final doubling. This method tested for the rate of change in the function to allow doubling when the points calculated became close enough, based on the DBLDF user input parameter.

The new method also allows halving of the step size when the points become too far apart. This information is stored in NSTPAR as a negative step count. For example: (NSTPAR(I),I=1,64) = 2 1 1 1 70 -9 -9 -9 -18 0 ... 0 tells us that after the fourth doubling, the step size halved three times for nine steps each halving, and then halved again for 18 steps at which point the function was effectively zero. In this method, the step size is not allowed to become less than the initial step size.

This modification was inserted easily into COVRGE without any drastic structural change to the program. The changes were tested to confirm that the program yields the same results for non-stress test cases. The FORTRAN 77 CASE1.DAT results (see figure 2-4) were exactly the same using both methods.

The change made to COVRGE enabling the adaptive integration step size to halve as well as double, was also tested using two stress test cases. The test cases were taken from the CARE III Phase III report (ref. 9). Test cases 3d and 3d'' were run. Figures 3-1 and 3-2 contain the list directed input files used which correspond to these two test cases:

```
1, 3.6E6, 3.6E3, 0.0, 3.6E3, 3.6E4, 1, 1, 1,,,, 0.01,,.TRUE./
1, 4, 2, 3, 2,,,,,3/
1, 1, 1.0, 1.0E-5/
60.0, 64, 2, .TRUE., .TRUE./
****TEST CASE-T3D  L.BRYANT  20OCT83****
 1 1 2 2
 2 0 1
CRITICAL PAIR FAULT TREE
 1 4 5 5
 1 1 4
 5 2 1 2 3 4
```

Figure 3-1 - List Directed Input File Test Case 3d

```
1, 3.6E6, 3.6E3, 3.6E4, 3.6E3, 3.6E4, 1, 1, 1,,,, 0.02,,.TRUE./
1, 4, 2, 3, 2,,,,,3/
1, 1, 1.0, 1.0E-5/
60.0, 64, 2, .TRUE., .TRUE./
****TEST CASE-T3D'' L.BRYANT  20OCT83****
 1 1 2 2
 2 0 1
CRITICAL PAIR FAULT TREE
 1 4 5 5
 1 1 4
 5 2 1 2 3 4
```

Figure 3-2 - List Directed Input File Test Case 3d''

Test case 3d was chosen because previously it was unable to run to completion due to accumulated error. It was hoped that the change to COVRGE would reduce the accumulated error, enabling it to run successfully. This was not the case however. The change did reduce the number of oscillations in the problem functions, specifically those functions used to compute single fault function P_a. The integration test for accumulated error of the kernel of function F_X must be less than or equal to 1.0 (see pages 11 and 12 in ref. 9). When the pro-

gram was run with the doubling difference input parameter DBLDF specified as 0.02, the integration error test result was 1.1322; with DBLDF specified as 0.01 (its minimum value) the integration result was 1.0217. Therefore this case was still unable to run to completion.

Test case 3d'' was chosen as a second stress case to run mainly to be able to compare the final result with the Phase III result. Test case 3d'' (see figure 3-2) was run using various versions of COVRGE and CARE3. Table 3-1 lists those results:

COVRGE Version	CARE3 Version	Final CARE3 Result
CDC COVRGE	CDC CARE3	2.6467251658E-12 *
VAX COVRGE	version V.4	1.4192107775E-12
VAX COVHLV **	version V.3	1.4189854803E-12
VAX COVHLV	version V.4	1.4189854803E-12

* From ref. 9;

** COVHLV is module COVRGE containing the step size halving capability.

Table 3-1 - Comparison of Final Result Using Various Versions of CARE III

The "halving step size" version of COVRGE effected the final result only slightly. The discrepancy between the original CDC version and the current VAX version results may be due to the numerous changes made to CARE III by Boeing Computer Services.

From what little testing has been done using the modified COVRGE program, it appears as if the halving capability yields only a minor improvement.

4.0 EXAMINATION AND ENHANCEMENT OF KOLMOGOROV FORWARD EQUATION SOLUTION

While reexamining the methods used to solve the CARE III Kolmogorov forward equation, it was determined that the original method of using NSTEPS equally spaced integration steps, in program CARE3, is valid only if FT (flight time) is short - for example, 10 hours. For a very large FT, all of the pertinent coverage information is contained within the first integration time step. Therefore to correct this deficiency, an integration step size method similar to that used in program COVRGE was implemented. This also caused changes to be made in program COVRGE because the moment functions must be passed to CARE3 at the same time points that the functions in CARE3 are to be calculated.

4.1 SOFTWARE DESIGN CHANGES

This section details the software design changes made to enable all reliability-model functions, in program CARE3, to be computed with increasing step sizes - similar to the method used in program COVRGE. It is not necessary to include a "halving step size capability" to module CARE3. The rational for using increasing step sizes is to start with a small step size required by the coverage-model functions and gradually increase to a large enough step size capable of reaching FT in 64 or less steps.

The design change that had to be addressed was how best to increase accuracy when solving the Kolmogorov forward equation, especially as FT (user specified flight or operating time) becomes large relative to the coverage-model functions. The change had to be made without completely restructuring CARE3 and without greatly increasing execution time and storage requirements. This method, in combination with a more sophisticated integration routine, yields greater accuracy when computing the $Q_{\ell}(t)$ vectors. All reliability-model functions computed using the first three moments of the coverage-model functions, i.e. all non-fault-vector-dependent functions, are computed with a greater concentration of points at the beginning of the time range where coverage is the gating factor. The details of the software design change follow.

4.1.1 REVIEW OF ORIGINAL SOFTWARE DESIGN

In the original version of CARE3, all reliability-model functions were computed at equal time steps, determined by the ratio 'FT / NSTEPS'. It is at these same times that the moments of the coverage-model functions were passed to CARE3. (See pg. 36 in ref. 8 for an explanation of the interface between COVRGE and CARE3 using the first three moments of the coverage-model functions.) As FT increases, fewer and fewer of the function values passed to CARE3 are unique since the first step often contains the entire coverage-model function. For example, if TZERO (time at which the corresponding

probability function became effectively zero) equals 15 minutes and FT equals 60 minutes, then 17 unique moment function values would be passed to CARE3. The function value at time zero, 15 values at multiples of time 0.9375 minutes, and the final function value would be passed leaving 'NSTEPS +1 -17' points passed that equal the 17th point value. The final function value is repeated to fill the remainder of the array passed to CARE3. If NSTEPS equals 64, then 48 duplicate points would be passed to CARE3. Table 4-1 shows NUVL (number of unique values passed to CARE3) as FT increases.

TZERO	FT	RELSTP	NUVL	
15 min.	1 hr.	0.9375 min.	17	(see text above)
15 min.	10 hr.	9.375 min.	3	i.e. value at time zero, 9.375 min., and the final value,
15 min.	100 hr.	93.75 min.	2	i.e. value at time 0.0 and the final value,
15 min.	1000 hr.	937.5 min.	2	
15 min.	5000 hr.	4687.5 min.	2	

where NUVL = TZERO/RELSTP + 1, RELSTP = FT/NSTEPS, and NSTEPS = 64.

Table 4-1 - Number of Unique Coverage Values Passed to CARE3 as FT Increases (Using Equal Step Sizes)

Table 4-1 shows that as FT increases, the entire coverage-model function is contained in the first RELSTP. It is represented as only two

values for CARE3 to work with when computing the convolution of the first three moments of the coverage-model function with the reliability-model weight function f(t) (see ref. 8, pg.42). As can be seen, FT does not have to become very large before this happens.

4.1.2 ENHANCED SOFTWARE DESIGN

In the enhanced version, the coverage functions are passed to CARE3 at time values determined by a "logarithmic time steps" method. This method is based on an initial coverage step size (CVGSTP), which is used for a number of steps, determined by the program, and then doubled. This doubled CVGSTP is then used for the same number of steps as the initial CVGSTP and then doubled again. This process continues until FT is reached because FT was the maximum time value requested for the current run.

4.1.2.1 COMPUTING THE DOUBLING PARAMETERS

Flight time can be described in terms of the doubling function using CVGSTP, NPERST and NDUB as follows:

$$FT = NPERST*CVGSTP*2^{0} + NPERST*CVGSTP*2^{1} + \ldots + NPERST*CVGSTP*2^{NDUB}$$

This equation can be simplified to the following one.

$$FT = NPERST * CVGSTP * (2^{NDUB+1} - 1)$$

The following list describes the variables in this equation:

1) FT - user defined flight or operating time for which the system is to be assessed,

2) NPERST - number of steps to use per each step size before doubling occurs,

3) CVGSTP - initial coverage step size,

4) NDUB - number of times CVGSTP must be doubled in order to reach FT.

The following constraints must be placed on NPERST, NDUB and CVGSTP when solving this "logarithmic time steps" equation.

- Let NSTEPS = NPERST * (NDUB+1). When varying NPERST and NDUB in the above equation - to solve for CVGSTP - NSTEPS must be computed less than or equal to 64. This restriction is necessary because module CARE3 uses a maximum of 64 steps for all function calculations.

- CVGSTP must be computed based on the behavior of the coverage functions in the current run. The most appropriate CVGSTP would be one based on the coverage-model probability function with the shortest time range. But this value would not necessarily yield valid integer values for NPERST and NDUB if plugged into the "logarithmic time steps" equation.

Because NSTEPS is still directly linked to the number of steps used to compute all the reliability functions in CARE3, NSTEPS continues to be a user specified parameter. NSTEPS is redefined as the minimum number of steps (not equally spaced) to be used in the COVRGE and CARE3 programs. If the original method of equally spaced time steps is desired, a new logical input parameter LGTMST (logarithmic time steps) may be input as .FALSE. in the input file. If .FALSE. is input for LGTMST, NSTEPS will be defined as it was originally.

A preliminary coverage step size (PCVSTP) is calculated as the minimum TZERO value of all the coverage-model probability functions divided by 64. PCVSTP is thus based entirely on the behavior of the coverage functions in the current run. Because PCVSTP would not necessarily be a valid solution to the "logarithmic time steps" equation, PCVSTP is used as a constraint when solving for CVGSTP. Because PCVSTP is the smallest appropriate initial coverage step size, PCVSTP must be used as the lower bound when computing CVGSTP.

4.1.2.2 DETERMINING A VALID SET OF CVGSTP'S

The "logarithmic time steps" equation can be rearranged, in order to solve for CVGSTP, as follows:

$$FT / CVGSTP = NPERST * (2^{NDUB+1} - 1).$$

FT can be eliminated from the equation, thus making the solution independent of FT, by introducing a new variable NCVSTP (number of initial CVGSTP's). NCVSTP equals the ratio 'FT / CVGSTP' in the equation, as shown below. Module COVRGE computes a set of NCVSTP values by iterating NPERST and NDUB when solving for NCVSTP in this equation.

$$NCVSTP = NPERST * (2^{NDUB+1} - 1).$$

Table 4-2 lists the possible values for NPERST and NDUB for NSTEPS values between 60 and 64. The final column NCVSTP (number of initial CVGSTP's), given NPERST and NDUB, will be used to determine

the best CVGSTP for the current coverage run. Note that the greatest expanse in the table occurs when 'NPERST = 1'. If the NSTEPS range is expanded to include a minimum value of 34, the table expands to a maximum of 63 solutions (not shown). A minimum value of 34 for NSTEPS assures the maximum number of choices for NCVSTP because the following two vector solutions, in the form (NPERST, NDUB, NSTEPS), yield the same NCVSTP value: (1, 32, 33) = (2, 31, 64).

NPERST	NDUB	NSTEPS	NCVSTP
1	63	64	.184467441E+20
1	62	63	.922337204E+19
1	61	62	.461168602E+19
1	60	61	.230584301E+19
1	59	60	.115292150E+19
2	31	64	.858993459E+10
2	30	62	.429496730E+10
2	29	60	.214748365E+10
3	20	63	6291453
3	19	60	3145725
4	15	64	262140
4	14	60	131068
5	11	60	20475
6	9	60	6138
7	8	63	3577
8	7	64	2040
9	6	63	1143
10	5	60	630
12	4	60	372
16	3	64	240
21	2	63	147
32	1	64	96
64	0	64	64

Table 4-2 - (NPERST, NDUB, NSTEPS) Vector Solutions Independent of FT

4.1.2.3 SELECTING THE MOST VALID CVGSTP

Now that a list of possible choices exist for CVGSTP, because FT and NCVSTP values are known and 'CVGSTP = FT / NCVSTP', the final step is to choose the most appropriate CVGSTP based on the behavior of the coverage functions in the current run. FT is divided by the preliminary step size (PCVSTP) and the result is compared to the NCVSTP choices. One NCVSTP is chosen such that it is the largest value less than or equal to 'FT / PCVSTP'. Choosing NCVSTP in this manner guarantees that CVGSTP will be greater than or equal to PCVSTP. CVGSTP can then be computed as 'FT / NCVSTP', and NPERST and NDUB values retrieved from the table.

All moments of the coverage-model functions are passed to CARE3 at times based on CVGSTP, NPERST and NDUB. The only additional information that must be passed to CARE3 are these three values: (CVGSTP, NPERST, NDUB), termed the doubling vector, which will enable module CARE3 to generate the time values which correspond to the passed coverage function values.

4.1.3 RESULTS OF ENHANCED SOFTWARE DESIGN

Table 4-3 shows the enhanced version's NUVL's versus the original version of COVRGE's NUVL's for a coverage function with a TZERO value of 15 minutes and NSTEPS specified as 60 (minimum):

FT (hr.)	NPERST	NDUB	NSTEPS	CVGSTP (min.)	ENHANCED NUVL	ORIGINAL NUVL
1	16	3	64	0.250	36	17
10	8	7	64	0.294	24	3
100	5	11	60	0.293	19	2
1000	4	14	60	0.458	14	2
5000	4	15	64	1.144	10	2
80000	3	20	63	0.763	10	2

Table 4-3 - Increased Number of Unique Coverage Values with Enhanced COVRGE

NPERST, NDUB and CVGSTP were computed using the method described above. Notice that using this method does not necessarily lead to fewer unique points as FT increases. This is because 'FT / PCVSTP' must fall very close to one of the possible NCVSTP values (see Table 4-2) in order to get CVGSTP as close to PCVSTP as possible. The 5000 hour and 80,000 hour cases both yield 10 unique function values before TZERO is reached because the 5000 hour CVGSTP was larger than the 80,000 hour one. If NSTEPS was input as 50 (minimum), the 5000 hour CVGSTP would equal 0.382 minutes with the solution vector (NPERST, NDUB, NSTEPS) = (3, 17, 54) and NUVL would increase to 13.

This example shows what will happen if NSTEPS is input as a value less than 64. CVGSTP will approach PCVSTP, NUVL will increase and NSTEPS will decrease. There is a trade-off between how many unique coverage function values to pass to CARE3 and how many time steps to use in computing the reliability functions in module CARE3. NSTEPS is still set to a default value of 50, which appears to satisfy COVRGE and CARE3 requirements for the best accuracy possible.

4.2 CODE CHANGES

The necessary code changes for this enhancement were kept to a minimum. The basic structure of both the COVRGE and CARE3 programs was not changed. The code that calculates the coverage-model probability functions was not modified. The "logarithmic time steps" method merely determines which corresponding moment function values to pass to CARE3.

Five new routines were added to COVRGE:

1) CIDXDV - compute index of doubling vector,

2) CKMNTS - check times of saved moment functions,

3) GNDBLV - generate array of possible doubling vector solutions,

4) GNLGTM - generate time array based on doubling parameters,

5) QDINTR - quadratic interpolation.

Routines FILSNG and FILDBL were expanded to fill the moment arrays at times based on the chosen doubling vector.

In CARE3, three new routines were also added:

1) CUBINT - Simpson's rule integration based on nonequally spaced abscissas,

2) FTMIDX - find time index,

3) RETFVL - retrieve weight function value,

SUBROUTINE UNRELQ was modified to use CUBINT instead of FINTGT; SUBROUTINE ABCST was modified to compute the a(t), b(t) and c(t) coefficients based on the initial step size CVGSTP at predetermined times instead of at multiples of the reliability step size.

The main change in the CARE3 program was the computation of all functions at times equivalent to the passed moments of the coverage functions. These times are computed using the passed doubling vector (CVGSTP, NPERST, NDUB) and stored in TMAR(65), which is located in COMMON /STEPCM/. When computing functions H_L, $H_{\bar{B}}$, H_B, H_{DPT}, h_{DF}, h_F, the reliability step size (RELSTP) was replaced by CVGSTP. (See Table 1, pp. 40-43 in ref. 8 for a description of the "H" functions.) And the integration method used to solve the Kolmogorov forward equation, which is comprised of the aforementioned functions, was changed to use a modified Simpson's rule, based on non-equally spaced abscissas.

More sophisticated integration routines, such as those based on Runge-Kutta formulas would not add accuracy to the calculation used to compute the "Q" vectors (see pg. 53 in ref. 5 and pg. 35 in ref. 8 for the definition of $Q_{\underline{\ell}}(t)$). Excessive interpolations, within the various stored functions that comprise the $Q_{\underline{\ell}}(t)$ definition, would be

required. The modified Simpson's rule integration routine uses the exact stored function values at non-equally spaced abscissas and thus does not require interpolations. That is why it was chosen over a Runge-Kutta method or a method that uses adaptive step sizes. There are too many preevaluated functions that comprise the calculation of the "Q" vectors to be able to use a numerical method that chooses the independent variable as it progresses.

4.3 TESTING OF THE CODE CHANGES

The coding of the design change to the Kolmogorov forward equation calculation, detailed in section 4.1, was debugged using Example Problem 5, contained in reference 1, with a modification to the PB input parameter in the intermittent fault type - set to 0.99 instead of 0.1. (The only reason for the change was that the test case was originally presented to Sequoia Systems, Inc. with 'PB = 0.99'.) A bug was found in the original code while debugging this change and will be discussed in this section.

The Example Problem 5 test case was chosen to aid in debugging the enhancement because of its varied fault types. In this enhanced method for solving the Kolmogorov equation, the unequal time steps used for solving all functions are based on the coverage functions' time maximums (TZERO points). This test case contained coverage functions with greatly varying TZERO values: 9.28E-6 hours to FT (10

hours). Therefore it was an excellent test case for debugging the generation of the doubling vector (CVGSTP, NPERST, NDUB), described in section 4.1, and its effect on the calculation of the failed state probabilities $Q_{\underline{\ell}}(t)$.

The calculation of the weight function f at time zero, used in the convolutional approximation (see Table 1 in ref. 8), was discovered to be incorrect. The error occurs mainly when the ω parameter, in the Weibull failure density function, is defined to be other that 1.0. The f(0) correction eliminates oscillation in the H_X (X = L, B, B, and DPT) and h_X (X = DF and F) functions, which directly affects the accuracy of the failed state probabilities $Q_{\underline{\ell}}(t)$.

The error in the calculation of f(0), using

$$\lambda_{X1}(0) = \omega_{X1}\,\lambda_{X1}^{\omega_{X1}-1}\,0$$

occurs when OMGA(ICAT,ISTG) is less than or equal to 1.0. Function f(0) is used extensively in calculating the a(t), b(t) and c(t) coefficients used in the convolution of the reliability-model function with the first three moments of the coverage-model functions. Therefore, in the original version of CARE III, if OMGA(ICAT,ISTG) is less than or equal to 1.0, the H_X functions oscillate slightly, which generates errors in the failed state probabilities $Q_{\underline{\ell}}(t)$.

Function FLAM, in module CARE3, calculates f(t) described above. It automatically set f(0) equal to zero, when actually the following three cases exist:

$$f(0) = \begin{cases} 0 & , \omega_{x1} > 1.0 \\ \lambda_{x1} & , \omega_{x1} = 1.0 \\ \infty & , \omega_{x1} < 1.0. \end{cases}$$

The correction to FUNCTION FLAM at time zero, when OMGA(ICAT,ISTG) is greater than or equal to 1.0, is straightforward:

```
FLAM = 0.0
IF (OMGA(ICAT,ISTG) .EQ. 1.0)  FLAM = RLAM(ICAT,ISTG)
```

Whereas the initialization and use of f(0) equal to infinity is impossible. Therefore the use of f(0) in the a(t), b(t) and c(t) computations had to be removed. J. J. Stiffler redefined these coefficients to be exact at one-sixth, midpoint and five-sixth of the range of interest of τ - instead of at the two end points and the midpoint - in order to eliminate the use of f(0) (see ref. 8, pg. 36). This correction entailed an extensive change to subroutine ABCST, in module CARE3 (see section 5.1).

Also included in this Kolmogorov equation enhancement was the computation of '1 - $R_x(t)$' - unreliability of a stage x module - using the series expansion: $\Lambda(t) - \frac{\Lambda^2(t)}{2} + \frac{\Lambda^3(t)}{6}$, where

$$\Lambda(t) = \sum_{1} \Lambda_{x1}(t) \quad \text{and} \quad \Lambda_{x1}(t) = \lambda_{x1}^{\omega_{x1}} t$$

in the instances where '1 - $R_x(t)$ < 1.0E-9' (see ref. 8, Table 1). For example, if $R_x(t)$ equals 0.999999999123456 (=1.0 single precision), '1 - $R_x(t)$' equals 0.0, while in fact '1 - $R_x(t)$' equals 0.876544E-9. Using the series expansion above gives a result accurate to eight or nine decimal digits without using double precision variables. This enhancement yielded better accuracy when computing $P^*_{\underline{\ell}}(t)$ - perfect coverage probability given failure vector $\underline{\ell}$ - which increased the accuracy of the failed state probabilities $Q_{\underline{\ell}}(t)$ (calculated using the Kolmogorov forward equation).

4.4 ENHANCEMENT RESULTS

The enhanced Kolmogorov forward equation calculation of the failed state probabilities $Q_{\underline{\ell}}(t)$ contains the following four positive features, which will be detailed below:

1) the "Q SUM" result (sum of failed state probabilities $Q_{\underline{\ell}}(t)$) is approximately one order of magnitude more accurate;

2) it is possible to track results from extremely small times through an extremely large FT;

3) it does not necessarily require an increase in execution time;

4) the vast majority of the CARE III code remained unchanged.

Tables 4-4, 4-5 and 4-6 contain comparison results of the enhancement, which uses a logarithmic type time scale (the initial step size CVGSTP is used NPERST times before it is doubled; this increased step size is used NPERST times before it is doubled; and this con-

tinues NDUB times before FT is reached), versus the equal step size option (FT divided by NSTEPS). (See section 4.1 for the detailed explanation of the enhancement design.) Results from a total of six runs, using Example Problem 5, are contained in Tables 4-4, 4-5 and 4-6. Three different FT values were used: 10 hours, 1600 hours and 80,000 hours. For each FT, two runs were executed - one using the logarithmic step size method (labeled "lg"), and one using equal time steps (labeled "eq"). Table 4-4 is a comparison of the six runs at time 10 hours; Table 4-5 is a comparison at 1600 hours - of the four runs containing this time; and Table 4-6 lists results at 80,000 hours using the two step size methods. Appendix B contains copies of the listings from these six runs of their final results.

FT (hr.)	Initial Step Size (lg or eq)	"Q SUM"	"P* SUM"	"Q+P* SUM"
10	0.2000e+0 (eq)	3.56506e-5	5.86195e-7	3.62368e-5
10	9.7704e-4 (lg)	3.58802e-5	5.86195e-7	3.64664e-5
1600	3.2000e+1 (eq)	2.22040e-5 +	5.86195e-7	2.27902e-5
1600	1.0173e-3 (lg)	3.58803e-5 *	5.86195e-7	3.64665e-5
80000	1.6000e+3 (eq)	1.05057e-6 +	5.86195e-7	0.16368e-5
80000	1.1921e-3 (lg)	3.58803e-5 *	5.86195e-7	3.64665e-5

\+
linear interpolation within first step
*
quadratic interpolation using three non-zero points

Table 4-4 - Example Problem 5 Results at 10 hours

FT (hr.)	Initial Step Size (lg or eq)	"Q SUM"	"P* SUM"	"Q+P* SUM"
1600	3.2000e+1 (eq)	1.44899e-3	1.37866e-2	1.52356e-2
1600	1.0173e-3 (lg)	1.48729e-3	1.37866e-2	1.52739e-2
80000	1.6000e+3 (eq)	1.68091e-4 *	1.37866e-2	1.39547e-2
80000	1.192le-3 (lg)	1.48344e-3	1.37866e-2	1.52700e-2

*
quadratic interpolation using three non-zero points

Table 4-5 - Example Problem 5 Results at 1600 hours

FT (hr.)	Initial Step Size (lg or eq)	"Q SUM"	"P* SUM"	"Q+P* SUM"
80000	1.6000e+3 (eq)	2.77075e-4	9.99658e-1	9.99935e-1
80000	1.192le-3 (lg)	1.57438e-3	9.99658e-1	1.00123e+0

Table 4-6 - Example Problem 5 Results at 80000 hours

The most valuable aspect of this enhancement is that the accuracy of the run no longer depends on FT and the number of steps requested. Compare the results at 10 hours in Table 4-4. Each of the enhancement runs yield the same result at 10 hours - including the 80,000 hour run, while the results steadily degrade using the equal

step size method. For very large FT, the "Q SUM" results - and thus the final unreliability - can be as much as one order of magnitude incorrect using the equal step size option. This can also be seen in the comparison at 1600 hours in Table 4-5. The fact that the CARE III model yields conservative failed state probabilities is exemplified in Table 4-6. The enhancement run declares that the unreliability at 80,000 hours is greater than 1.0.

It is no longer necessary to make multiple runs using various FT's to try to achieve valid results at different times. Only one run is necessary using the maximum desired time as FT. It is possible to see accurate results at extremely small times through extremely large times with the one run. If FT is modified in the input file CREIN and program CAREIN is rerun, it is still of the upmost importance that program COVRGE be rerun before rerunning program CARE3. The COVRGE functions are passed to CARE3 at times based on the coverage functions' TZERO points and FT.

This enhancement does not in itself cause an increase in execution time because all functions are computed at the same number of times steps as the equal step size method. But in order to generate any failed state probabilities for runs with extremely large FT's, the truncation test for $Q_{\underline{\ell}}(t)$ probabilities, based on the PSTRNC input parameter, had to be modified. This modification results in a larger number of "Q" vectors being computed, which does increase the execution time. The previous truncation test was based on the

value of $P^*_{\underline{\ell}}(t)$ at FT. If '$P^*_{\underline{\ell}}(FT)$ < PSTRNC', the corresponding "Q" vector was not computed. For very large FT, the perfect coverage probability, given a failure vector $\underline{\ell}$, may equal zero at FT even though the failed state probabilities for vector $\underline{\ell}$ may be large enough to effect the final result. The enhanced truncation test is currently based on the maximum $P^*_{\underline{\ell}}(t)$ value given failure vector $\underline{\ell}$. This results in the opposite problem that too many "Q" vectors are computed. It was necessary to increase the PSTRNC input parameter default value from 1.0E-14 to 1.0E-10 to minimize this problem. In the original released version of CARE III, it is possible that some important failed state probabilities are missing owing to the original PSTRNC truncation test.

From these test cases, it appears as if this enhancement has added not only more accuracy to the failed state probabilities but has solved most of the problems with the large FT runs.

5.0 ELIMINATION OF OSCILLATION IN MODULE CARE3 FUNCTIONS

SUBROUTINE ABCST, in module CARE3, was recoded and throughly debugged during this enhancement phase. It no longer calls FUNCTION FLAM for time zero. The need for this change was explained in section 4.3. The details of this change will be discussed in section 5.1. Several test cases were run, and their ensuing problems, solutions and results will be detailed in section 5.2.

5.1 CODE CHANGES RELATED TO FUNCTION OSCILLATION PROBLEMS

The following mathematical expressions redefine the solution of the a(t), b(t) and c(t) coefficients used in the convolution of the reliability-model function with the coverage-model functions. (See the CARE III Phase II report (ref. 8), pg. 36 for a discussion of this convolution.) J. J. Stiffler redefined a(t), b(t) and c(t) to make the approximation $p_1(t-\tau) = a(t) + \tau b(t) + \tau^2 c(t)$ exact at known points that are as close as possible to the one-sixth, one-half and five-sixth points in the range of interest of τ. This eliminates the use of the weight function:

$$f(t) = \lambda_{x1}(t) = \omega_{x1}\,\lambda_{x1}\,t^{\omega_{x1}-1}$$

at time zero, which equals infinity when ω_{x1}, OMGA(ICAT,ISTG), is less than 1.0. This redefinition of a(t), b(t) and c(t) revolves around the Kolmogorov forward equation calculation enhancement discussed in section 4.0 and its subsections. The independent variable

in the weight function, used to solve for a(t), b(t) and c(t), is no longer a multiple of Δt_r, with Δt_r the reliability step size 'FT/NSTEPS'. The time chosen as the independent variable in the weight function is now based on times generated by the doubling vector (CVGSTP, NPERST, NDUB) as discussed in section 4.1.

The mathematical expressions used to recode SUBROUTINE ABCST follow.

Given:

$$t_0 = \begin{cases} t_{max} & \text{for the minimum value of } t \text{ for which} \\ & p_X(t) \le \theta \text{ or } P_X(t) \le \theta \text{ for all } t > t_{max}, \\ & \text{with } \theta \text{ a user-defined threshold,} \\ FT & \text{for } p_X(t) \text{ or } P_X(t) \text{ that reach a non-zero} \\ & \text{steady-state value,} \end{cases}$$

where $p_X(t)$ (X = DF and F) and $P_X(t)$ (X = L, B, $\overline{B}$, and DPT) are the coverage-model functions, and FT is the user-defined flight time or operating time.

$t_r = \min(t, t_0)$

t_1 = closest stored point to $t - 5/6\ t_r$

t_2 = closest stored point to $t - 1/2\ t_r$

t_3 = closest stored point to $t - 1/6\ t_r$

where $0 < t_1 < t_2 < t_3$.

Choosing t_1, t_2 and t_3 in this manner eliminates the need for interpolation within the weight function f(t). (See ref. 8, pg. 42 for the complete definition of f(t).)

a(t), b(t) and c(t) are defined as follows:

$$a(t) = \frac{(t-t_2)(t-t_3)}{(t_2-t_1)(t_3-t_1)} f(t_1) - \frac{(t-t_1)(t-t_3)}{(t_2-t_1)(t_3-t_2)} f(t_2) + \frac{(t-t_1)(t-t_2)}{(t_3-t_1)(t_3-t_2)} f(t_3)$$

$$b(t) = -\frac{2t-t_2-t_3}{(t_2-t_1)(t_3-t_1)} f(t_1) - \frac{2t-t_1-t_3}{(t_2-t_1)(t_3-t_2)} f(t_2) + \frac{2t-t_1-t_2}{(t_3-t_1)(t_3-t_2)} f(t_3)$$

$$c(t) = \frac{1}{(t_2-t_1)(t_3-t_1)} f(t_1) - \frac{1}{(t_2-t_1)(t_3-t_2)} f(t_2) + \frac{1}{(t_3-t_1)(t_3-t_2)} f(t_3)$$

The above change to SUBROUTINE ABCST, in module CARE3, yields the same results, up to the sixth decimal digit, as the previous method for 'OMGA(ICAT,ISTG) >= 1.0'. And it eliminates the oscillation problem with the H_X (X = L, B, $\overline{B}$, and DPT) and h_X (X = DF and F) functions when 'OMGA(ICAT,ISTG) < 1.0'. Several runs, discussed below, will illustrate this fact.

5.2 TESTING AND RESULTS OF CODE CHANGES

The following seven test cases, contained in Table 5-1, were run to test this change to SUBROUTINE ABCST. They are listed in the order in which they were run and are discussed below.

Test Case	Name	ω	Source
1	transient 2c	1.0	reference 9.
2	transient 2c	0.8	modification to Test Case 1.
3	Example Problem 5	1.0	reference 1.
4	Example Problem 8	1.0	reference 1.
5	permanent 1e*	0.5	reference 9.

* ρ = 3.6E3 (misprint in ref. 9, pg. 120 has ρ = 3.6E4)

6	permanent 1f	2.0	reference 9.
7	permanent 1i	1.0	reference 9.

Table 5-1 - Elimination of Function Oscillation Test Cases

Test cases 1 and 2, i.e. transient 2c with differing OMGA (ICAT,ISTG) values, were run before SUBROUTINE ABCST was modified. Test case 1 was run to see if the correction to FUNCTION FLAM, discussed in section 4.3, eliminated the oscillation problem in the "P* SUM" and thus the "Q+P* SUM" results. That is exactly what happened. (See ref. 9, pp. 42, 43 and 121 for the earlier results.) Originally the oscillation in the "P* SUM" results was caused by the following function:

$$R_{x1}(t) = e^{-H_{DPT}(t|x1)} .$$

(Note that this is a correction to the misprint in ref. 8, pg. 37.

Also the definition of $H_{DPT}(t|x1)$ on pg. 41 should have capital M's to represent the moment functions.) The $R_{x1}(t)$ function was used in the calculation of the $P^{*}_{\ell}(t)$ function. The H_{DPT} function contained oscillations previously due to the error in FUNCTION FLAM at time zero, discussed in section 4.3, which returned zero instead of λ_{x1}, RLAM(ICAT,ISTG), for 'OMGA(ICAT,ISTG) = 1.0'. These oscillations have been totally removed. The total system unreliability increased from 9.7825E-16 to 3.8337E-15 at 60 minutes, due to the corrections and enhancements.

Test case 2 was run with 'OMGA = 0.8' to be used as a comparison before and after SUBROUTINE ABCST was modified to eliminate FUNCTION FLAM at time zero. Before the correction, function $H_{DPT}(t|x1)$ oscillated due to the incorrect use of FLAM, at time zero equal to zero, in SUBROUTINE ABCST. Just as in the first test case, the total system unreliability increased - in this case from 6.3681E-13 to 4.3350E-12 at 60 minutes. This increase was due solely to the correction to SUBROUTINE ABCST.

Test case 3, i.e. Example Problem 5, was run to check that the new SUBROUTINE ABCST performed as well as the prior version for a run without the oscillation problem, i.e. a run with 'OMGA = 1.0'. This test case was used earlier to debug the Kolmogorov forward equation calculation enhancement. The results prior to the ABCST change were presented in section 4.3. The "Q SUM" results are equivalent to the earlier run up to the sixth decimal digit. This

is a direct consequence of the same amount of difference existing in the H_X and h_X functions between the two runs. Therefore, it was concluded that the new version of ABCST performs at least as well - probably better due to the elimination of the interpolation within the weight function - as the previous version for runs with 'OMGA (ICAT,ISTG) >= 1.0'. It now performs correctly for runs with 'OMGA (ICAT,ISTG) < 1.0'.

Test case 4, i.e. Example Problem 8, was run mainly to compare to the results contained in reference 1, using the enhancement code on a run with critical pairs. This test case consists of two subruns. The first subrun does not contain critical pair data in file "BXYFL" but the second subrun does. Yet when this test case was run, the following message was printed for both subruns: ** Warning - Critical Fault Pair File "BXYFL" does not contain data for this subrun. **. After a considerable amount of debugging was performed, the error was tracked to SUBROUTINE CRTLPR, in module CAREIN, and SUBROUTINE BUFBLK, contained in all three modules of CARE III. When the CDC version of CARE III was converted to run on the VAX machine, the block buffering commands BUFFER IN and BUFFER OUT were incorrectly converted using DO loops. This caused one machine word records to be written (read) to (from) disk instead of using large blocks of words per record. This error resulted in three problems:

1) BACKSPACE 14 statement in SUBROUTINE NFLTDP, in module CARE3, repositioned file "BXYFL" incorrectly for runs with critical pair data in other than the first subrun; (Runs with critical pair data in the first subrun only executed properly because file "BXYFL" did not have to be repositioned.)

2) increased disk file sizes by 200 percent;

3) increased I/O accesses by 225 percent in module CAREIN, 16 percent in module COVRGE and 31 percent in module CARE3.

In SUBROUTINE CRTLPR, the critical pair data is supposed to be written to file "BXYFL" in blocks of 15,000 words per subrun (if critical pairs exist for that subrun). Using one word records caused the BACKSPACE 14 statement to backspace one word instead of one 15,000 word block. When the second subrun tried to read the critical pair data, it hit end-of-file after one word and assumed that no critical pair data existed. This error was temporarily corrected, in SUBROUTINE NFLTDP, by putting the BACKSPACE 14 statement in a 15,000 count DO loop. This enabled the run to execute correctly albeit very inefficiently. Statistics were taken on the file sizes, execution times and I/O accesses for this run. The incorrect conversion code was then corrected by replacing the DO loops with one implied DO statement. The following example was taken from SUBROUTINE BUFBLK:

Incorrect Conversion of BUFFER IN:

```
      DO 110 IL = 1,ILST
         READ(IUNIT,END=300,ERR=400) BLOCK(IL)
110   CONTINUE
```

Correct Conversion of BUFFER IN:

```
      READ(IUNIT,END=300,ERR=400) (BLOCK(IL),IL=1,ILST)
```

This type of correction was made to routines CRTLPR (in module CAREIN), BUFBLK (in all three modules) and BUFDAT (in module CARE3). The 15,000 count BACKSPACE 14 DO loop was removed from SUBROUTINE NFLTDP; the run was executed again and the same statistics were taken as for the previous run. The conversion error resulted in the 200 percent increase in disk file sizes because the UNIX operating system adds two words per record to the file for the end-of-record indicator and system use. The increased number of I/O accesses, caused by the incorrect conversion of the BUFFER IN and BUFFER OUT statements, yielded the following unnecessary increases in execution time:

CAREIN: one word records - 174.9 seconds (1158% increase)
blocked records - 13.9 seconds

COVRGE: one word records - 1117.6 seconds (16% increase)
blocked records - 962.9 seconds

CARE3: one word records - 1056.8 seconds (57% increase)
blocked records - 672.6 seconds.

Both versions of CARE III (blocked and unblocked) yielded the exact same results. The total system unreliability equaled 5.8807E-7 in reference 1 for Example Problem 8 and equaled 5.8968E-7 using the enhanced version of CARE III.

Test cases 5, 6 and 7, i.e. permanent test cases 1e, 1f and 1i taken from reference 9, were run to further test the SUBROUTINE ABCST correction for 'OMGA < 1.0' and the enhancement code. Table 5-2 is a list of the results for the three test cases as reported in the CARE III Phase III report (ref. 9) versus the results using the enhanced version of CARE III.

Test Case	Name	ω	λ	Version 3 "Q+P* SUM"	Enhanced "Q+P* SUM"
5	1e *	0.5	1.0E-10	2.3101E-13	4.4851E-13
6	1f	2.0	3.162E-3	2.4821E-13	2.4820E-13
7	1i	1.0	1.0E-5	1.8629E-13	1.8725E-13

* ρ = 3.6E3 (misprint in ref. 9, pg. 120 has ρ = 3.6E4)

Table 5-2 - Permanent Test Case Results at 60 min.

As was expected, test case 6 currently yields basically the same result as version 3. This is because FUNCTION FLAM and SUBROUTINE ABCST performed correctly for 'OMGA > 1.0' (= 2.0 in this case). Test case 7 results differ slightly due to the previously discussed error (see section 4.3) where FLAM returned zero at time zero, when 'OMGA = 1.0', instead of RLAM (= 1.0E-5 in this case). The greatest difference occurred in test case 5, as was expected, due to the prior incorrect use of FUNCTION FLAM at time zero, in SUBROUTINE ABCST, when 'OMGA < 1.0'. The current results now behave as expected when comparing the three cases to each other. The failure rates in the three cases are shown in figure 5-1.

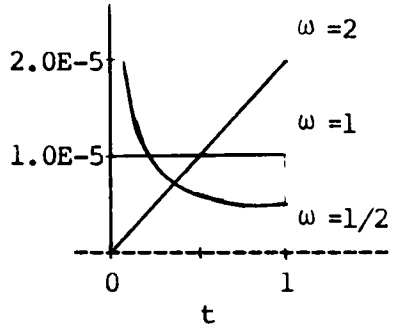


Figure 5-1 - Graph of Failure Rates for Test Cases 5, 6 and 7

The average failure rate is the same in all three cases - but the 'OMGA = 2' and 'OMGA = 1/2' cases both result in more clustered failures ('OMGA = 1/2' is clustered at 't < 1/4'; 'OMGA = 2' is clustered at 't > 1/2') - since the dominate cause of failure is presumably due to the critical pairs, both 'OMGA = 2' and 'OMGA = 1/2' should be worse than 'OMGA = 1'. Further since the 'OMGA = 1/2' failure rate for 't < 1/16' is greater than it ever gets for 'OMGA = 2' (since the maximum value of t in this case is 1 hour), 'OMGA = 1/2' should give the poorest results.

To summarize - the oscillation problem in the various H_X and h_X functions has been eliminated, and SUBROUTINE ABCST has been corrected and recoded to fit into the Kolmogorov enhancement code.

6.0 CONSOLIDATION OF BOEING'S VERSION 4 AND ENHANCED CARE III CODE

Boeing Computer Services' final version of CARE III (ref. 6) was converted to FORTRAN 77 and the enhancements detailed thus far inserted into that version. The additional code needed to pass the TZERO times (maximum times for which the coverage functions are non-negligible) to CARE3 was added to Boeing's exponential coverage model. This oversite in the Boeing code is described below.

6.1 CODE CONVERSION AND INSERTION

The rewritten CAREIN module, received from NASA with the Boeing FTREE routines removed, was converted to FORTRAN 77 standard. This was a quick task - the only required change was the elimination of the NAMELIST's. All required changes to the CAREIN module, starting with the internally redundant modules enhancement, were made only to this new version of CAREIN.

Concerning program COVRGE, Boeing's coverage routines and changes were successfully merged with the enhancement coverage code. Boeing's coverage routines converted to FORTRAN 77 readily due to their using six characters or less variable names.

Each separate coverage function's TZERO point must be passed to CARE3. These times are passed to CARE3 in arrays TZROST(5,5), for coverage single-fault functions, and TODF(5,5), for coverage double-fault functions. These arrays are contained in COMMON block /CVRGCM/.

While looking at the output of the COVRGE runs resulting from Boeing's changes to include an alternate exponential coverage model, it was evident that the TZERO times were missing on the printouts, i.e. they were zero for all non-zero functions. Boeing may have been under the misconception that these times are computed in CARE3 (see pg. 105 in ref. 5). This is incorrect. It is not possible to compute the TZERO point of the probability function from the corresponding moments that are passed to CARE3. Routines FHSFST, FHDFST, ABCST, FFSFST and FFDFST in CARE3 will not execute properly without these TZERO times. Note that a given function's TZERO value will equal FT if the function reached a non-zero steady state value.

The following code was added to Boeing's MSNGFN and MDBLFN subroutines in module COVRGE and tested. This code sets maximum function time values into arrays TZROST(ITYP,MCHI) and TODF(ITYP,JTYP), which are contained in common area /CVRGCM/. This common area was also added to both subroutines. Module CARE3 uses these values when convolving the moments of the coverage functions with the reliability function.

Insertion into SUBROUTINE MSNGFN:

```
      .
      .
      .
C
C     *** STEADY STATE REACHED ***
C
  150 CONTINUE
C
      KSNG = ITM
```

```
C
C     WRITE (6, 9020) ITYP, ITM, T1
C
C     *** STORE MAXIMUM TIME FOR WHICH EACH COVERAGE FUNCTION     ***
C     *** IS NONNEGLIGIBLE.  MODULE CARE3 USES THIS VALUE WHEN    ***
C     *** CONVOLVING THE COVERAGE FUNCTIONS WITH THE RELIABILITY ***
C     *** FUNCTION.                                               ***
C
      DO 155 I = 1, 5
C
C        *** IN THE MAJORITY OF CASES, FUNCTION IS NEGLIGIBLE ***
C        *** AFTER 'T1'                                       ***
C
         TZROST( ITYP, I) = T1
C
C        *** TEST FOR ZERO VALUED FUNCTION OR FUNCTION THAT   ***
C        *** REACHED A NON-ZERO STEADY STATE VALUE            ***
C
         FKI = FSNG( KSNG, I)
         IF (FKI .EQ. 0.0) THEN
C
C           *** ZERO VALUED FUNCTION ***
C
            TZROST( ITYP, I) = 0.0
C
         ELSE
C
            FKIM1 = FSNG( KSNG-1, I)
            FKIM2 = FSNG( KSNG-2, I)
            IF (FKI.GT.0.0 .AND. FKIM1.GT.0.0 .AND. FKIM2.GT.0.0) THEN
               FMAX = AMAX1(FKI, FKIM1, FKIM2)
C
C              *** FUNCTION REACHED NON-ZERO STEADY STATE VALUE? ***
C              *** IF SO, IT IS NONNEGLIGIBLE THROUGH 'FT'       ***
C
               IF ((ABS(FKI - FKIM2) / FMAX) .LE. 1.E-4)
     .             TZROST( ITYP, I) = FT
C
            ENDIF
C
         ENDIF
C
  155 CONTINUE
C
      GO TO 170
C
C     *** NUMERICAL INTEGRATION ERROR ***
C
      .
      .
      .
```

Insertion into SUBROUTINE MDBLFN:

```
      .
      .
      .
C
C     *** STEADY STATE REACHED ***
C
  150 CONTINUE
C
      KDBL = ITM
C
C     WRITE (6, 9020) ITYP, JTYP, ITM, T1
C
C     *** STORE MAXIMUM TIME FOR WHICH EACH COVERAGE FUNCTION     ***
C     *** IS NONNEGLIGIBLE.  MODULE CARE3 USES THIS VALUE WHEN    ***
C     *** CONVOLVING THE COVERAGE FUNCTIONS WITH THE RELIABILITY  ***
C     *** FUNCTION.                                               ***
C
C     *** IN THE MAJORITY OF CASES, FUNCTION IS NEGLIGIBLE        ***
C     *** AFTER 'T1'                                              ***
C
      TODF( ITYP, JTYP) = T1
C
      FKIJ = FDBL( KDBL)
      IF (FKIJ .EQ. 0.0) THEN
C
C        *** ZERO VALUED FUNCTION ***
C
         TODF( ITYP, JTYP) = 0.0
C
      ELSE
C
         FKIJM1 = FDBL( KDBL-1)
         FKIJM2 = FDBL( KDBL-2)
         IF (FKIJ.GT.0.0 .AND. FKIJM1.GT.0.0 .AND. FKIJM2.GT.0.0) THEN
            FMAX = AMAX1(FKIJ, FKIJM1, FKIJM2)
C
C           *** FUNCTION REACHED NON-ZERO STEADY STATE VALUE? ***
C           *** IF SO, IT IS NONNEGLIGIBLE THROUGH 'FT'       ***
C
            IF ((ABS(FKIJ - FKIJM2) / FMAX) .LE. 1.E-4)
     .          TODF( ITYP, JTYP) = FT
C
         ENDIF
C
      ENDIF
C
      GO TO 170
C
C     *** NUMERICAL INTEGRATION ERROR ***
C
      .
      .
      .
```

The enhancement discussed in section 4.0 and its subsections was successfully inserted into Boeing's exponential coverage model; two related bugs were found in Boeing's coverage model; and Boeing's final version of CARE3 was successfully converted to FORTRAN 77 and compiled cleanly.

The "logarithmic time step" option (set user parameter LGTMST = .TRUE.) code was readily inserted into Boeing's exponential coverage model. Routines MSNGMT and MDBLMT, which compute the moments of the single and double-fault output functions respectively, were changed to integrate at times determined by the doubling parameters CVGSTP, NPERST and NDUB instead of at multiples of RELSTP. The integration subroutine (HSGEAR) performed as well at the unequal time intervals as it did using equal time steps.

CARE3 was converted to FORTRAN 77 and compiled cleanly. All seven character variable names were converted to six characters - as was done with the original version of CARE3 sent to Sequoia Systems, Inc. in August, 1983. The code had been completely restructured by Boeing in the areas where the Kolmogorov equation enhancements had been made. Thus all Kolmogorov equation enhancements had to be restructured to fit into the new version.

A debug subroutine PRNTFN was added to the CARE3 module. This subroutine writes all formerly buffered H_X (X = L, B, $\overline{B}$ and DPT) and h_X (X = DF and F) functions to a text file. This file was used to check the Boeing conversion as well as the enhancement additions.

The following is a list of the major enhancement changes made to the converted FORTRAN 77 version of Boeing's CARE3 module. See the referenced sections for a detailed description of each enhancement.

1) All functions are now computed using times stored in TMAR(ITSTPS) instead of at multiples of RELSTP (see sections 4.2 and 4.4).

2) The $1 - R_X(t)$ enhancement calculation using the series expansion was added in all appropriate routines (see section 4.3).

3) FUNCTION FLAM was corrected at time zero when OMGA equals 1.0 (see section 4.3).

4) FUNCTION FINTGT, Simpson's rule integration routine based on equally spaced abscissas, was replaced by CUBINT, Simpson's rule integration based on nonequally spaced abscissas (see section 4.2).

5) SUBROUTINE UNRELQ was changed to integrate the SUMK array using SUBROUTINE CUBINT (see section 4.2).

6) FUNCTION FRXIFF was changed to integrate the h_{DPT} function using SUBROUTINE CUBINT. (Boeing conversion change to integrate h_{DPT} function in CARE3 instead of in module COVRGE).

7) SUBROUTINE ABCST was replaced with the enhanced version that does not require the weight FUNCTION FLAM at time zero (see section 5.1).

8) SUBROUTINE BUFDAT and SUBROUTINE BUFBLK were changed to buffer blocks of data as opposed to one word buffers (see section 5.2).

9) SUBROUTINE PRNTFN was added to print functions resulting from the convolution of the single and double-fault coverage functions with the reliability-model function. These functions are contained in COMMON /BXYCOM/ and are written to file PRFNCS (unit 13), if CARE3 is compiled with CHIDBG set .TRUE. (New subroutine comparable to Boeing removed PRNTGH subroutine. See section 8.3 for a description of the functions stored in COMMON block BXYCOM).

The plot programs CVGPLT and RELPLT were upgraded to interface correctly with the enhanced version of CARE III. The COMMON blocks were upgraded to match those in programs COVRGE and CARE3, and the plot programs were converted to FORTRAN 77. The data structure of the plotting files is thoroughly detailed in sections 8.1 and 8.2.

6.2 TESTING OF CONSOLIDATED CODE

Example Problem 8, taken from reference 1, was used to test the enhancement code insertion into the exponential coverage model. The resulting moment functions were compared to output results using 'MARKOV = 2', i.e. the general coverage model, and 'LGTMST = .TRUE.'. They compared extremely well for moment functions produced from the integration of steadily decreasing functions. The moment functions that did not compare were those produced by integrating functions that increased to a steady state value. The reason for this poor comparison is discussed in section 6.3.

Example Problem 8, taken from reference 1, was also used to test the CARE3 consolidation. The "Total System Unreliability" result at 10 hours, using 'LGTMST = .TRUE.', equaled 5.8820228332E-7. The "P" and "Q" vector results looked fine but functions in file PRFNCS for Subrun 2 only were totally incorrect. All functions in Subrun 1 were perfect but the functions in Subrun 2 appeared as if some important data was overwritten between subruns. This code error is discussed in section 6.4.

6.3 CODE ERRORS FOUND DURING COVRGE CONSOLIDATION

Two related bugs were found in Boeing's code while debugging the enhancement. SUBROUTINE HSGEAR produces integrated moment functions which oscillate starting at the fourth or fifth decimal digit. Since the moment functions are the result of an integration, they should never decrease. Initially the oscillations were blamed on the enhancement code, but the oscillations were also found in output listings sent to Sequoia Systems, Inc. by NASA in December, 1983. Further investigation proved that the oscillations are inherent in Boeing's original code but become more obvious with the enhancement style of integration using smaller step sizes. Removing the C's from column one (comment line indicator) on the Boeing debug code WRITE statements, and running Example Problem 8, shows that the functions being integrated to become the moment functions are not always well behaved. One oscillates from positive to negative and back to positive values. Many simply decrease steadily and abruptly become negative. But even well behaved functions produce integrated functions that oscillate slightly.

The second bug occurs while integrating functions that reached a non-zero steady state value. The integration is not performed correctly through FT thus producing a much smaller moment function. For example, function PBNG in fault type 4 of Example Problem 8 reaches a steady state value of 0.9897 at time 0.1842E-2 hours. The function remains at this value through 'FT = 10 hours', yet the

first moment function integration result at 10 hours is 0.1531E-2. The general coverage model integrates through FT and yields the correct result at 10 hours of 9.897.

Both problems are the result of Boeing's code integrating the functions farther in time than they were computed and using a zero value, i.e. using 'f(> tmax) = 0'. This is fine for functions that decreased to zero, but not for functions that reached a non-zero steady state value. The code was changed to return the last computed value of the function, when results were requested at a point farther in time than computed. This increased the oscillation problem, and the integrated function went negative when integrating a function that went negative - although it did correct the steady state valued function integration result. Both the equal and non-equal step size runs yielded the correct result of 9.897 at 10 hours for fault type 4, moment zero function MBNG.

The code change was then corrected to use the final function value, during integration, only for functions that reached steady state. Also added was code to remove the oscillations from the moment functions after they are computed. The maximum integration result is repeated through FT, starting at the first decrease in the moment function. This time corresponds to the maximum time that the coverage function was computed.

6.4 CODE ERRORS FOUND DURING CARE3 CONSOLIDATION

Four bugs were found while debugging the Boeing and Sequoia consolidation of the CARE3 module. Two bugs were found in the new Boeing CARE3 code, one in the original CARE3 code, and one in the enhancement code in module COVRGE. These four bugs and their solutions will be described below.

The four bugs were discovered using the debug SUBROUTINE PRNTFN, mentioned in section 6.1, which writes all H_X (X = L, B, $\overline{B}$ and DPT) and h_X (X = DF AND F) functions to text file PRFNCS as they are computed. Since these functions are the link between the coverage model and the reliability model, it is important that these functions are computed as accurately as possible. Each bug is described using a three step approach - describe the symptom of the bug, the problem caused by the bug, and the correction made to the code to correct the bug.

The first bug affects the single-fault calculations while the remaining three bugs affect the double-fault calculations. The four bugs can be summarized as follows:

1) Garbage in H_X and h_X functions computed in subruns 2 and above.

2) Double-fault arrays XXDFP and XXDFT, calculated using the h_{DF} function and the reciprocal of the H_L function, too small becoming more accurate as the function progresses in time.

3) Oscillation in the $\begin{Bmatrix} XX \\ XY \\ YX \end{Bmatrix}$DFP and $\begin{Bmatrix} XX \\ XY \\ YX \end{Bmatrix}$DFT double-fault arrays.

4) Double-fault arrays, listed above, equal to zero for several initial time steps then abruptly become large positive values.

Description of the first bug:

SYMPTOM:

Erratic values were generated for functions H_X (X = L, B, $\bar{B}$ and DPT) and h_X (X = F) in subruns greater than one, in module CARE3. For example, the integration of function h_{DPT} went from 0.27E-14 at time 4.47E-6 (8th time step) to 0.0 at time 6.85E-6 (9th time step) to 0.17E-11 at time 9.23E-6 (10th time step).

PROBLEM:

The single-fault coverage moment functions, stored in arrays CMST0(5,65,5), CMST1(5,65,5) and CMST2(5,65,5) in COMMON /CVRGCM/, were overwritten by the "P* SUM" function calculation between subruns.

CORRECTION:

In the CARE3 main program, Boeing incorrectly substituted array PSTFAR for array SRPSTF in the subrun "P*" function calculation. Since PSTFAR is equivalenced to CMST0(1,1,1) in COMMON /CVRGCM/, the single-fault coverage functions were overwritten with the "P*" calculation before the second subrun. The "P*" calculation code was returned to its original form using array SRPSTF. Array SRPSTF

(65,20) was defined and placed in COMMON /BXYCOM/ after array AXAR (20,65) and equivalenced to AXAR(1,1), in order not to increase storage requirements. In the original code, SRPSTF was contained in COMMON /NONLDP/ and equivalenced to AXIAR(1,1,1). Boeing removed COMMON /NONLDP/ and converted AXIAR(5,20,65) to AXAR(20,65) when they removed the buffering to disk of the H_X and h_X functions. Therefore the "P* SUM" calculation is now correct and functionally equivalent to the original design, and the coverage moment functions remain intact between subruns.

Description of the second bug:

SYMPTOM:

The h_{DF} functions, computed in SUBROUTINE GNTXX, contained much smaller values in the XXDFP and XXDFT arrays than was correct when compared to the results computed by hand. The XXDFP array contains values obtained by taking the reciprocal of the H_L function squared, multiplied by the h_{DF} function at a given time. Therefore it was possible to check the program's results using the debug print file PRFNCS.

PROBLEM:

The XXDFP and XXDFT arrays were computed using a constant value for the H_L function. Boeing used an uninitialized variable XHLTP to represent the H_L function within the double-fault time step loop. Since this same variable had been used previously in the single-fault functions' calculations, it contained the H_L function value at FT. Therefore the H_L function was treated as a constant function during the calculation of the double-fault functions.

CORRECTION:

In SUBROUTINE GNTXX, the following line of code was inserted after the DO 230 IT = 1, ITSTPS line: XHLTP = HLTP(ISTG, IT). This initializes XHLTP to the H_L function value that corresponds to the current time step.

Description of the third bug:

SYMPTOM:

The double-fault functions' calculations, stored in arrays $\begin{Bmatrix} XX \\ XY \\ YX \end{Bmatrix}$DFP(530,1) (double-fault permanent) and $\begin{Bmatrix} XX \\ XY \\ YX \end{Bmatrix}$DFT(530,1) (double-fault transient) in COMMON /BXYCOM/, contained slight oscillations. The functions would slightly increase then decrease repeatedly as the function approached FT.

PROBLEM:

The weight function f(t), used in the convolutional approximation when computing the h_{DF} functions, was programmed originally using an incorrect definition. The incorrect code uses $r_x(t)$ when x_1 is a transient fault. The following is the correct definition:

$$f(t) = H_B(t|x_1) \lambda_y(t) \begin{cases} r_x(t)\, r_y(t) & : x_1 \text{ and } y_1 \text{ are non-transient faults} \\ r_x(t) & : x_1 \text{ non-transient and } y_1 \text{ transient faults} \\ r_y(t) & : x_1 \text{ transient and } y_1 \text{ non-transient faults} \\ 1 & : x_1 \text{ and } y_1 \text{ are transient faults} \end{cases}$$

CORRECTION:

In FUNCTION FFDFST, in module CARE3, the sense of the transient fault test was incorrect. The test was corrected to read:

IF(.NOT.TRNSFC(ICAT,ISTG)) HBNGVR = HBNGVR * RXAR(ISTG,KINDX).

This has the effect of taking the H_B function and multiplying it by $r_x(t)$, if x is a non-transient fault. The previous result is then multiplied by λ_{yj} (t) and by $r_y(t)$, if y is a non-transient fault. This correction removed the slight oscillations contained in the double-fault arrays as the functions approached FT.

Description of the fourth bug:

SYMPTOM:

The double-fault function arrays, listed in the previous bug description, contained zeros for several initial time steps then abruptly took on relatively large values.

PROBLEM:

The double-fault moment functions, passed from the COVRGE module, were incorrect for the first several time steps. The functions were linear at the beginning due to COVRGE using linear interpolation for too many points within the first time step of the double-fault coverage functions. This was done because of the enhancement code scheme to use the minimum TZERO value of all coverage functions to compute the CVGSTP - coverage step size for passing the moment functions to module CARE3.

CORRECTION:

Module COVRGE was modified to use the maximum TZERO value, less than FT, of all coverage functions to compute the CVGSTP. This totally eliminates interpolation within the first step of any coverage function, where the function tends to change most rapidly. The effect that this correction has on the enhancement code is to generate a much larger CVGSTP, relative to the original method (see section 4.1), but it is still many orders of magnitude smaller than the 'RELSTP = FT / NSTEPS' method used when 'LGTMST = .FALSE.'. Therefore the enhanced method, used when 'LGTMST = .TRUE.', is still as valid as originally designed, and the resulting moment functions that are passed to CARE3 are more accurate.

7.0 IMPLEMENTATION OF INTERNALLY REDUNDANT MODELING CAPABILITY

There are two possible methods for representing internally redundant modules in the CARE III model. The straightforward approach is simply to break each internally redundant module into one or more separate stages and to treat each of its constituent submodules as modules in one of those stages. Consider, for example, a memory module consisting of 40 bit lines supported by four redundant bit lines and some common logic needed to control memory access and to effect reconfiguration should one of the bit lines fail. This module could be treated as two stages, one stage representing the common logic and the second the 44 bit lines configured in an "m-of-n" configuration (with m = 40 and n = 44).

The disadvantage of this approach is that both the number of stages and the number of modules that have to be accommodated by the model may have to be considerably greater than they would have been were the modules not internally redundant. If, in the previous example, the system contains one stage with ten such internally redundant modules, using the approach just described would increase the number of stages to 20 (10 representing the 10 sets of common logic and 10 representing the 10 sets of 44 bit lines) and the number of modules would increase from 10 to 450 (10 + 44*10). This of course, could vastly increase the computational time and could even exceed the program's capacity.

An alternative method is to recognize that each such internally redundant module appears to the system as a module subject to transient failures. That is, each failure of a redundant submodule is a transient event in that it incapacitates the module only temporarily so long as a redundant submodule is available and recovery is successful. Thus, the module can be modeled by defining a transient failure rate in terms of the rate at which submodule failures occur until no redundant submodules remain, and a permanent failure rate in terms of the rate at which failures occur either in the non-redundant portion of the module or in the redundant portion when no redundant submodules remain. The coverage model then determines the probability that the system recovers from each of these two types of failures. (Note that the coverage parameters need not be the same for the two failure types.)

7.1 MATHEMATICAL MODEL

Specifically, let (λ_1, ω_1) be the Weibull parameters characterizing the failure rate of each of the redundant submodules and let (λ_2, ω_2) be the failure rate parameters for the non-redundant portion of the module. Further, let n be the number of available submodules and m the minimum number needed for the module to function and consider the following two cases:

Case 1 - REDUNDANT SUBMODULES INACTIVE -

Only active submodule failures cause reconfiguration. The internally redundant module always uses exactly m submodules to perform its intended function. Each of the remaining functioning submodules is used only as needed to replace one of these submodules (or an earlier replacement for one of these submodules). Thus, reconfiguration is undertaken only when one of the m currently used submodules fails. The internally redundant module fails when there is no longer a redundant submodule to replace an active, failed submodule. The model assumes that when one of the redundant submodules replaces a failed submodule, it will have the same failure rate as the other functioning submodules. This is true, for example, if the redundant submodules are always powered or in any other situation in which the dormancy factor is unity.

Example: Random access memory devices in combination with an error detecting code. When an error is detected, recovery involves testing the memory to isolate the defective device and switching in a replacement. Until that time, the replacement devices are off-line; a failure in one of these devices may be detected by a background test but does not cause a reconfiguration.

Case 2 - REDUNDANT SUBMODULES ACTIVE -

Any submodule failure forces a reconfiguration. The internally redundant module uses all available submodules. Each failure results in a reconfiguration with the reconfigured internally redundant module continuing to function with successively fewer submodules until the number of functioning submodules drops below m.

Example: N-modular redundancy with an adaptive voter. Each non-unanimous vote causes a reconfiguration in which all submodules not agreeing with the majority are switched out.

Using input parameter ACSP(x), defined in section 7.2, the user chooses which case to model: Case 1 (REDUNDANT SUBMODULES INACTIVE) or Case 2 (REDUNDANT SUBMODULES ACTIVE). Note that Case 1 is more reliable than Case 2. The reason for this is that in Case 1, only m submodules can fail and, by failing, cause a reconfiguration, thereby making the system vulnerable to a recovery failure. In Case 2, a failure of any of the n submodules can result in a system failure.

The submodule transient failure rate $\lambda_1(t)$ and the module permanent failure rate $\lambda_2(t)$ can be defined as follows. (Note that $\lambda_1(t)$ is a function of time and uses the Weibull parameters (λ_1, ω_1) in its definition; likewise $\lambda_2(t)$ is a function of time and uses the Weibull parameters (λ_2, ω_2). See pp. 29-30 in ref. 2 for a description of the Weibull parameters.)

$$\lambda_1(t) = \begin{cases} \dfrac{m\,\omega_1\,\lambda_1\,t^{\omega_1-1}\,t^{\omega_1-1}\sum_{\ell=0}^{n-m-1} P_\ell^*(t)}{\sum_{\ell=0}^{n-m} P_\ell^*(t)} & \text{Case 1} \\ \dfrac{\omega_1\,\lambda_1\,t^{\omega_1-1}\,t^{\omega_1-1}\sum_{\ell=0}^{n-m-1} (n-\ell)\,P_\ell^*(t)}{\sum_{\ell=0}^{n-m} P_\ell^*(t)} & \text{Case 2} \end{cases}$$

$$\lambda_2(t) = \frac{m\,\omega_1\,\lambda_1\,t^{\omega_1-1}\,t^{\omega_1-1}\,P_{n-m}^*(t)}{\sum_{\ell=0}^{n-m} P_\ell^*(t)} + \omega_2\,\lambda_2\,t^{\omega_2-1}\,t^{\omega_2-1} \qquad \text{Cases 1 and 2}$$

$$e^{-\Lambda_2(t)} = \sum_{\ell=0}^{n-m} P_\ell^*(t)$$

with

$$P^{*}_{\ell}(t) = \binom{n}{\ell}\left(e^{-(\lambda_1 t)^{\omega_1}}\right)^{n-\ell}\left(1 - e^{-(\lambda_1 t)^{\omega_1}}\right)^{\ell} e^{-(\lambda_2 t)^{\omega_2}}$$

Function $e^{-\Lambda_2(t)}$ is used in the computation of the function $R_x(t)$: reliability of a stage x module. It specifies the probability that a given stage x module, with internally redundant submodules, has not experienced a permanent type fault by time t. (See ref. 8, pg. 37 for the equivalent definition for modules with no internal redundancy.)

7.1.1 DERIVATION OF MATHEMATICAL MODEL

Derivation:

$\lambda_1(t)$ = submodule failure rate at time t given that the internally redundant module will still be operational if it successfully recovers from the failure (i.e. that at least m operational submodules remain after the failure).

Thus $$\lambda_1(t) = \sum_{\ell=0}^{n-m-1} \lambda_1(t|\ell)\, P_{\ell}(t|S)$$

with $\lambda_1(t|\ell)$ - the submodule failure rate given that the internally redundant module has suffered ℓ previous submodule failures

and $P_{\ell}(t|S)$ - the probability that the internally redundant module has suffered ℓ failures by time t given that it is still operational at time t.

Because only active submodule failures are relevant in Case 1 while all submodule failures are relevant in Case 2:

$$\lambda_1(t|\ell) = \begin{cases} m\,\omega_1\,\lambda_1^{\omega_1}\,t^{\omega_1-1} & \text{Case 1} \\ (n-\ell)\,\omega_1\,\lambda_1^{\omega_1}\,t^{\omega_1-1} & \text{Case 2} \end{cases}$$

Further

$$P_{\ell}(t|S) = \frac{P_{\ell}^{*}(t)}{\sum_{\ell=0}^{n-m} P_{\ell}^{*}(t)}$$

$e^{-\Lambda_2(t)}$ = probability that the internally redundant module is still operational at time t given no coverage failures

$$= \sum_{\ell=0}^{n-m} P_{\ell}^{*}(t)$$

$$= \sum_{\ell=0}^{n-m} \binom{n}{\ell} p_1^{n-\ell}(t)\,(1 - p_1(t))^{\ell}\, p_2(t)$$

with

$$p_1(t) = e^{-(\lambda_1 t)^{\omega_1}}$$

$$p_2(t) = e^{-(\lambda_2 t)^{\omega_2}}$$

$\lambda_2(t)$ = internally redundant module failure rate at time t

$$= \left(\frac{d}{dt} e^{-\Lambda_2(t)}\right) \Big/ e^{-\Lambda_2(t)}$$

$$= \sum_{\ell=0}^{n-m} \sigma_\ell(t)\, \omega_1 \lambda_1^{\omega_1} t^{\omega_1 - 1}$$

$$+ \sum_{\ell=0}^{n-m} P^*_\ell(t)\, \omega_2 \lambda_2^{\omega_2} t^{\omega_2 - 1}$$

with $\sigma_{\ell}(t) = \binom{n}{\ell} p_1^{n-\ell}(t)\ (1 - p_1(t))^{\ell-1}$

$$* \left[(n-\ell)\ (1 - p_1(t)) - \ell\, p_1(t)\right] p_2(t)$$

But

$$(n - \ell)\binom{n}{\ell} = n\binom{n-1}{\ell}$$

$$\ell \binom{n}{\ell} = n\binom{n-1}{\ell-1}$$

so that

$$\sum_{\ell=0}^{n-m} \sigma_{\ell}(t) = n \sum_{\ell=0}^{n-m} \binom{n-1}{\ell} p_1^{n-\ell}(t)\ (1 - p_1(t))^{\ell}\ p_2(t)$$

$$- n \sum_{\ell=0}^{n-m-1} \binom{n-1}{\ell} p_1^{n-\ell}(t)\ (1 - p_1(t))^{\ell}\ p_2(t)$$

$$= n\binom{n-1}{n-m} p_1^{m}(t)\ (1 - p_1(t))^{n-m}\ p_2(t)$$

$$= m\, P^{*}_{n-m}(t)$$

the last expression following from the equality

$$n\binom{n-1}{n-m} = m\binom{n}{n-m}.$$

7.1.2 LIMITATION OF MATHEMATICAL MODEL

This second approach to modeling internally redundant modules should not result in any significant increase in computational time due to this internal redundancy. The approach does suffer from one limitation, however: it is not possible to treat the case in which submodules are critically coupled (e.g., the case in which one submodule fails during recovery from an earlier submodule failure and thereby causes a system failure). If the submodules are critically coupled in this way, the first of the two approaches should be used instead (see section 7.0).

7.2 ADDED INPUT PARAMETERS

The following input parameters, listed with their corresponding NAMELIST and definition, were added to module CAREIN to accommodate the internally redundant modeling capability:

NAMELIST	PARAMETER	DEFINITION
$STAGES	NSUB(x)	Number of identical submodules within each internally redundant module in stage number x. The default value for NSUB(x) is 0.
$STAGES	MSUB(x)	Minimum number of identical submodules needed within each internally redundant module in stage number x. The default value for MSUB (x) is 0.

NAMELIST	PARAMETER	DEFINITION
$STAGES	ACSP(x)	Flag set .TRUE. states that all non-failed submodules are used within each internally redundant module in stage number x (redundant submodules active). If set .FALSE., only MSUB(x) submodules are active at any instant (redundant submodules inactive). The default value for ACSP(x) is .TRUE.
$FLTCAT	JSBTYP(i,x)	Defines the fault type(s) assigned to stage x, with internally redundant modules, that affect the redundant submodules. Note that for stage x, with internally redundant modules, the original fault type parameter JTYP (i,x) defines the fault type(s) assigned to stage x that affect either the non-redundant portion of the module or the redundant portion when no redundant submodules remain. The default value for JSBTYP(i,x) is 1.
$FLTCAT	OMGSUB(i,x)	Parameter ω of the Weibull fault occurrence rate $\lambda\omega(\lambda t)^{\omega-1}$ characterizing the failure rate of each of the redundant submodules for fault type i for stage x with internally redundant modules. Note that for stage x, with internally redundant modules, the original ω parameter OMG(i,x) defines the Weibull parameter ω characterizing the failure rate of either the non-redundant portion of the module or the redundant portion when no redundant submodules remain. The default value for OMGSUB(i,x) is 1.0.
$FLTCAT	RLMSUB(i,x)	Weibull parameter λ characterizing the failure rate of each of the redundant submodules for fault type i for stage x with internally redundant modules. Note that for stage x, with internally redundant modules, the original λ parameter RLM(i,x) defines the Weibull parameter λ characterizing the failure rate of either the non-redundant portion of the module or the redundant portion when no redundant submodules remain. The default value for RLMSUB(i,x) is 1.0E-4.

Some differing terminology exists between this document and the "CARE III Model Overview and User's Guide (first revision)", reference 2, pertaining to modeling faults using CARE III. This document contains the original terms used when CARE III was first coded because of the numerous references to the program variables and their corresponding terms. The following list details the differing terminology.

	Original Terminology	Reference 2 Terminology	Defined By Input Parameters	Using NAMELIST
1)	fault type	fault model	ALP(1), BET(1), DEL(1), etc.	$FLTTYP
2)	fault category	fault type	RLM(1,x), RLMSUB(1,x), etc.	$FLTCAT

For each stage with internally redundant modules, up to five different pairs of fault occurrences (or categories) may be defined. Each such fault pair consists of a permanent fault category associated with either the non-redundant portion of the module or the redundant portion when no redundant submodules remain - defined using parameters JTYP(1,x), QMG(1,x) and RLM(1,x) - and a second permanent fault category associated with each of the redundant submodules - defined using parameters JSBTYP(1,x), QMGSUB(1,x) and RLMSUB(1,x). Thus for a stage x with internally redundant modules, up to ten different types of fault occurrences may be defined but they must be defined in pairs - one permanent fault category, used in the module permanent failure rate function $\lambda_2(t)$, and a second permanent fault category, used in the submodule transient failure rate function $\lambda_1(t)$.

Note that original parameter NFCATS(x) defines the number of pairs of fault categories assigned to stage x with internally redundant modules. Thus NFCATS(x) must always be defined less than or equal to five.

The coverage fault types assigned to JTYP(1,x) and JSBTYP(1,x), for stage x with internally redundant modules, must have ALP(1) and BET(1) parameters set equal to zero (the default value), which defines the fault-handling model type as a permanent one. The submodule transient failure rate function $\lambda_1(t)$ accounts for the transient faults resulting from submodule failures, in accordance with the user-defined permanent, submodule fault category. This fault category can be termed the "effective transient" fault category.

7.3 CODE ADDITIONS AND MODIFICATIONS

The CAREIN module was modified to accept the new input parameters. The CARE3 module was modified to accept and use the new input parameters to calculate the failure probability of a system defined with stage(s) consisting of internally redundant modules. It was not necessary to modify the fault tree handling portion of module CAREIN; therefore the CAREIN changes were straightforward and minimal. Tests were added to SUBROUTINE VLDNML to check the correctness of the new input parameters. Module COVRGE was not effected by this enhancement and thus was not modified. Module CARE3 required quite a few additions and modifications to handle this enhancement.

The modifications made to module CARE3 revolve around the computation of the reliability functions H_X (X = L, B, and $\overline{B}$) and h_X (X = DPT, F and DF). Additions were made to routines GNTXX(ISTG) and GNTXY(ISTG) to compute all functions over the fault category pairs described above, for ISTG with internally redundant modules. The logical array TRNSFC(i,x) used to test for a transient or non-transient fault category i for stage number x was converted to an integer array NTRNFC(i,x) with the following definition:

$$NTRNFC(i,x) = \begin{cases} 0 & \text{specifies a non-transient fault category,} \\ 1 & \text{specifies a transient fault category,} \\ 2 & \text{both a permanent and an "effective transient" fault category i exist for stage x with internally redundant modules.} \end{cases}$$

It was necessary to add a 6500 word array to COMMON /BXYCOM/ - HBNGSB(20, 5, 65). It was not possible to equivalence HBNGSB to another array. Array HBNGSB indexed by (x, i, t) is paired with original array HBNG(20, 5, 65) also indexed by (x, i, t). For stage number x with internally redundant modules HBNG contains the H_B function for the permanent fault category i given stage number x, while the HBNGSB array contains the H_B function for the "effective transient" fault category i given stage numoer x. This was the only additional array required for this enhancement because the H_B function is the only function stored per fault category. All other H_X and h_X functions are stored as sums over the transient and non-transient fault categories. Thus the fault category pairs defined for stage number x with internally redundant modules were summed into

the appropriate variables. Variables ending in P received the permanent fault category contribution, and variables ending in T received the "effective transient" fault category contribution.

Slight modifications were made to FUNCTIONS FFSFST and FFDFST to check which fault category was currently being processed of the pair of fault categories, for ISTG with internally redundant modules. Given fault category (ICAT,ISTG) for single-fault functions, logical variable TRNISB set .TRUE. flags the "effective transient" fault category (set .FALSE. flags the permanent fault category) of the fault category pair as the one currently being processed. For double-fault functions, TRNISB is used for the fault category (ICAT, ISTG) and TRNJSB is used for (JCAT,JSTG). Logical variables TRNISB and TRNJSB were added to COMMON /CONFIG/ after array NTRNFC(i,x) defined above.

Additions were made to FUNCTION FLAM to compute the module transient failure rate $\lambda_1(t)$ and the module permanent failure rate $\lambda_2(t)$, mathematically defined in section 7.1, for ISTG with internally redundant modules. FUNCTION FPSTSB was added to module CARE3 to compute the $P_\ell^*(t)$ function, also defined in section 7.1, which is used in FUNCTION FLAM to compute the module failure rates.

7.4 TESTING OF THE ENHANCEMENT

The original FTMP test case, used to test Phases I, II and III of the original CARE III program (refs. 8 and 9), was used to debug

this enhancement. The memory modules' stage was additionally defined with NSUB equal to 44 and MSUB equal to 40 with redundant submodules inactive.

Several different FTMP test cases were generated to test the internally redundant modeling capability. The main two test cases that helped debug this enhancement are contained in figures 7-2 and 7-3. The first consists of the original test case with no internal redundancy, and the second contains internal redundancy in the memory modules but has the submodule failure rate set to zero. These two test cases execute different code paths and routines, yet must yield exactly the same final results. After several iterations of testing and bug corrections, that is exactly what happened.

The two test cases described above are listed along with the original FTMP test case (fig. 7-1) for comparison. The FTMP critical pairs fault tree is not shown in all cases because it is lengthy and was not modified at all for these test runs. The input is shown in both list directed format and in NAMELIST format. Notice that a third permanent fault type was added. This was necessary because of the internal redundancy test case. Stages with internally redundant modules must have fault categories defined using only permanent fault types. In order to get the exact same final results, the run without internal redundancy had to have the memory modules' stage's fault categories defined using the same fault type as the run with the internally redundant memory modules.

FTMP Original Test Case:

List-directed Syntax:

```
2,     10.0,       10.0,
        1.0,        1.0,
      100.0,      360.0,
        0.0,        0.0,
        0.0,        0.0,
          1,          1,
          1,          1,
          1,          1,
        1.0,        1.0,
        1.0,        1.0,
        1.0,        1.0,
       0.05,     1.0E-5,
       .TRUE.,   ,   ,
       1, .TRUE./
3,       15,     9,     5,
         11,     5,     3,
           ,   ,   ,   ,   ,
           ,   ,   ,   ,   ,
          3,   ,   ,   ,   ,
           ,   ,   ,   4/
2,  2,  1,
 ,   ,
1,  1,
2,
 ,   ,
 ,   ,
 ,
1.0E-4,  0.18E-4,
1.0E-4,  0.18E-4,
1.0E-6/
100.0,  50,  2,
.TRUE.,  .TRUE.,  ,
1.0E-10/
```

NAMELIST Syntax:

```
$FLTTYP  NFTYPS=2,
         ALP   = 2 * 10.,
         BET   = 2 * 1.,
         DEL   = 100., 360.,
         RHO   = 2 * 0.,
         EPS   = 2 * 0.,
         IDELF= 2 * 1,
         IRHOF= 2 * 1,
         IEPSF= 2 * 1,
         PA    = 2 * 1.,
         PB    = 2 * 1.,
         C     = 2 * 1.,
         DBLDF = 0.05,
         TRUNC = 1.0E-5,
         CVPRNT = .TRUE.,
         MARKOV = 1,
         LGTMST = .TRUE.$
$STAGES  NSTGES = 3, N = 15, 9, 5,
                     M = 11, 5, 3,
         NOP(1,3) = 3, IRLPCD = 4$
$FLTCAT  NFCATS = 2, 2, 1,
         JTYP(1,2) = 2 * 1,
         JTYP(1,3) = 2,
         RLM(1,1) = 1.0E-4,
         RLM(2,1) = 0.18E-4,
         RLM(1,2) = 1.0E-4,
         RLM(2,2) = 0.18E-4,
         RLM(1,3) = 1.0E-6$
$RNTIME  FT = 100.0, NSTEPS = 50,
         ITBASE = 2,
         SYSFLG = .TRUE.,
         CPLFLG = .TRUE.,
         PSTRNC = 1.0E-10$
```

(FTMP Original Test Case continued)

```
 FTMP ARCHITECTURE - 15 PROCESSORS, 9 MEMORY MODULES, 5 BUSES
 WITH CRITICAL FAULT PAIRS (ORIGINAL TEST CASE) - NOVEMBER, 1984.
 1 3 4 4
 4 0 1 2 3
 FTMP CRITICAL-FAULT PAIRS FAULT-TREE.
 1 29 30 55
 1 1 15
 2 16 24
 3 25 29
30  2  1 2 3
31  2  4 5 6
32  2  7 8 9
33  2  10 11 12
34  2  13 14 15
35  0  30 31 32 33 34
36  2  16 17 18
37  2  19 20 21
38  2  22 23 24
39  0  36 37 38
40  2  25 26 27 28 29
41  0  2 3 5 6 8 9 11 12 14 15
42  A  25 41
43  0  1 3 4 6 7 9 10 12 13 15
44  A  26 43
45  0  1 2 4 5 7 8 10 11 13 14
46  A  27 45
47  0  42 44 46
48  0  17 18 20 21 23 24
49  A  25 48
50  0  16 18 19 21 22 24
51  A  26 50
52  0  16 17 19 20 22 23
53  A  27 52
54  0  49 51 53
55  0  35 39 40 47 54
```

Figure 7-1 - FTMP Original Test Case (pre-internally redundant modeling capability format)

List-directed Syntax:

```
3,   10.0,    10.0,     0.0,
      1.0,     1.0,     0.0,
    100.0,   360.0,   100.0,
      0.0,     0.0,     0.0,
      0.0,     0.0,     0.0,
        1,       1,       1,
        1,       1,       1,
        1,       1,       1,
      1.0,     1.0,     1.0,
      1.0,     1.0,     0.0,
      1.0,     1.0,     1.0,
     0.05,  1.0E-5,
     .TRUE.,  ,  ,
     1, .TRUE./
3,     15,       9,       5,
       11,       5,       3,
         ,        ,        ,
         ,        ,        ,
         ,        ,        ,
         ,  ,  ,  ,  ,
         ,  ,  ,  ,  ,
        3,  ,  ,  ,  ,
         ,  ,  ,  4/
2,  2,  1,
 ,  ,
3,  3,
2,
 ,  ,
 ,  ,
 ,
 ,  ,
 ,  ,
 ,
1.0E-4,  0.18E-4,
1.0E-4,  0.18E-4,
1.0E-6,
 ,  ,
 ,  ,
 ,
 ,  ,
 ,  , /
100.0,  50,  2, .TRUE.,  .TRUE.,  ,
1.0E-10/
```

NAMELIST Syntax:

```
$FLTTYP  NFTYPS=3,
         ALP  = 2 * 10., 0.0,
         BET  = 2 *  1., 0.0,
         DEL  = 100., 360., 100.,
         RHO  = 3 * 0.,
         EPS  = 3 * 0.,
         IDELF= 3 * 1,
         IRHOF= 3 * 1,
         IEPSF= 3 * 1,
         PA   = 3 * 1.,
         PB   = 2 * 1., 0.,
         C    = 3 * 1.,
         DBLDF = 0.05,
         TRUNC = 1.0E-5,
         CVPRNT = .TRUE.,
         MARKOV = 1,
         LGTMST = .TRUE.$
$STAGES  NSTGES = 3, N = 15, 9, 5,
                     M = 11, 5, 3,
         NOP(1,3) = 3, IRLPCD = 4$
$FLTCAT  NFCATS = 2, 2, 1,
         JTYP(1,2) = 2 * 3,
         JTYP(1,3) = 2,
         RLM(1,1) = 1.0E-4,
         RLM(2,1) = 0.18E-4,
         RLM(1,2) = 1.0E-4,
         RLM(2,2) = 0.18E-4,
         RLM(1,3) = 1.0E-6$
$RNTIME  FT = 100.0, NSTEPS = 50,
         ITBASE = 2,
         SYSFLG = .TRUE.,
         CPLFLG = .TRUE.,
         PSTRNC = 1.0E-10$
```

```
FTMP ARCHITECTURE - 15 PROCESSORS, 9 MEMORY MODULES, 5 BUSES
WITH CRITICAL FAULT PAIRS (NO INTERNAL REDUNDANCY) - NOVEMBER, 1984.
1 3 4 4
4 0 1 2 3
.
.  (Critical Pair Fault-Tree not listed)
.
```

Figure 7-2 - FTMP Modified Test Case (without internally redundant modules)

List-directed Syntax:

```
3,  10.0,    10.0,     0.0,
    1.0,     1.0,      0.0,
  100.0,   360.0,    100.0,
    0.0,     0.0,      0.0,
    0.0,     0.0,      0.0,
      1,       1,        1,
      1,       1,        1,
      1,       1,        1,
    1.0,     1.0,      1.0,
    1.0,     1.0,      0.0,
    1.0,     1.0,      1.0,
   0.05,  1.0E-5,
 .TRUE.,  ,  ,
      1, .TRUE./
3,   15,       9,        5,
     11,       5,        3,
      0,      44,        0,
      0,      40,        0,
 .TRUE.,  FALSE.,   .TRUE.,
       ,  ,  ,  ,  ,
       ,  ,  ,  ,  ,
      3,  ,  ,  ,  ,
       ,  ,  ,  4/
2,  2,  1,
 ,  ,
3,  3,
2,
 ,  ,
3,  3,
 ,
 ,  ,
 ,  ,
 ,
1.0E-4, 0.18E-4,
1.0E-4, 0.18E-4,
1.0E-6,
 ,  ,
 ,  ,
 ,
 ,  ,  0.0, 0.0, /
100.0,  50,  2, .TRUE.,  .TRUE.,  ,
1.0E-10/
```

NAMELIST Syntax:

```
$FLTTYP  NFTYPS=3,
         ALP   = 2 * 10., 0.0,
         BET   = 2 *  1., 0.0,
         DEL   = 100., 360., 100.,
         RHO   = 3 * 0.,
         EPS   = 3 * 0.,
         IDELF= 3 * 1,
         IRHOF= 3 * 1,
         IEPSF= 3 * 1,
         PA    = 3 * 1.,
         PB    = 2 * 1., 0.,
         C     = 3 * 1.,
         DBLDF = 0.05,
         TRUNC = 1.0E-5,
         CVPRNT = .TRUE.,
         MARKOV = 1,
         LGTMST = .TRUE.$
$STAGES  NSTGES = 3,
         N = 15, 9, 5,
         M = 11, 5, 3,
         NSUB = 0, 44, 0,
         MSUB = 0, 40, 0,
         ACSP = .T., .F., .T.,
         NOP(1,3) = 3, IRLPCD = 4$
$FLTCAT  NFCATS = 2, 2, 1,
         JTYP(1,2) = 2 * 3,
         JTYP(1,3) = 2,
         JSBTYP(1,2) = 2 * 3,
         RLM(1,1) = 1.0E-4,
         RLM(2,1) = 0.18E-4,
         RLM(1,2) = 1.0E-4,
         RLM(2,2) = 0.18E-4,
         RLM(1,3) = 1.0E-6$
         RLMSUB(1,2) = 0.0,
         RLMSUB(2,2) = 0.0$
$RNTIME  FT = 100.0, NSTEPS = 50,
         ITBASE = 2,
         SYSFLG = .TRUE.,
         CPLFLG = .TRUE.,
         PSTRNC = 1.0E-10$
```

```
FTMP ARCHITECTURE - 15 PROCESSORS, 9 MEMORY MODULES, 5 BUSES
WITH CRITICAL FAULT PAIRS AND INTERNALLY REDUNDANT MEMORY MODULES.
NOVEMBER, 1984 - With RLM = ORIGINAL LAMBDA and RLMSUB = 0.0.
THIS SHOULD YIELD THE EXACT SAME RESULTS AS THE TEST CASE NOT USING
INTERNALLY REDUNDANT MODULES.
1 3 4 4
4 0 1 2 3
.
.  (Critical Pair Fault-Tree not listed)
.
```

Figure 7-3 - FTMP Modified Test Case (with internally redundant memory modules)

The final result for both test cases at 100.0 minutes equaled 1.1187474946E-08 exactly. (See Appendix C for the summary results of the run using the test case shown in figure 7-3. This listing also shows the modified output style due to the internally redundant modeling enhancement.)

While running these test cases, a bug was found in Boeing's coverage model in program COVRGE. When COVRGE was run with input parameter MARKOV set to 1, the resulting double-fault moment functions were exactly zero. When the general fault-handling model was run, the resulting double-fault moment functions contained valid non-zero values. The error was tracked down to subroutine MDBLFN in module COVRGE. Two integer variables LAM1 and LAM2 were being used as real variables, i.e. they should have been declared real in a TYPE statement. A computed real value based on the input parameters BET, DEL and RHO was stored in LAM1 and LAM2. In the FTMP test case, this computed value equaled $0.0166\overline{6}$ minutes and when placed in the integer variables, was truncated to zero - thus the zero valued double-fault function results. These two variables were renamed RLAM1 and RLAM2, in subroutine MDBLFN, and the program ran correctly.

Four additional variations on the FTMP modified Test Case (with internally redundant memory modules) - see figure 7-3 - were run to test the internally redundant modeling capability. The internally redundant portion of the computer configuration, modeled in this test case, has nine memory modules in stage two. Each memory module con-

sists of 40 bit lines supported by four redundant bit lines, and some common logic needed to control memory access and to effect reconfiguration should one of the bit lines fail. A minimum of five memory modules are required for this stage to be operational. The input parameters used to describe this configuration for stage two are N(2) = 9, M(2) = 5, NSUB(2) = 44, MSUB(2) = 40, and ACSP(2) = .FALSE.

The Weibull input parameters (RLMSUB(1,2), OMGSUB(1,2)) characterizing the transient failure rate of each of the active bit lines, and (RLM(1,2), OMG(1,2)) characterizing the permanent failure rate of the common logic were varied, in the first three test cases listed in Table 7-1, such that '(MSUB * RLMSUB(1,2)) + RLM(1,2)' equals the original failure rate used in the FTMP test case without internally redundant memory modules. Two fault categories were defined for the memory module stage, as in the original test case: the original failure rate for the first fault category equaled 1.0E-4 and for the second fault category equaled 0.18E-4. Therefore originally 'RLM(1,2) = 1.0E-4' and 'RLM(2,2) = 0.18E-4' with 'OMG(1,2) = OMG (2,2) = 1.0', which says that each memory module is susceptible to a permanent fault with either constant rate 1.0E-4 per hour or 0.18E-4 per hour.

The fourth test case, listed in Table 7-1, was discussed above. The failure rates were defined so that the unreliability of the system would equal exactly the FTMP test case run without internal redundancy in the memory modules. By setting the failure rate parameter

'RLMSUB(1,2) = 0.0', which says that the 44 submodules per memory module never fail, eliminates any transient failure contribution by the redundant portion of the memory modules, and thus gives the same results as the test case defined without internally redundant memory modules.

The fifth test case, listed in Table 7-1, is defined so that both RLM(1,2) and 'MSUB * RLMSUB(1,2)' equal the original failure rates. That is, the common logic in each memory module is susceptible to a permanent fault with either constant rate 1.0E-4 per hour or 0.18E-4 per hour, and each active bit line is susceptible to a transient fault with either constant rate 2.5E-6 per hour or 4.5E-7 per hour.

The total system unreliability results for each test case described above is shown in Table 7-2. Note that the test cases have been ordered by increasing unreliability results. Also notice that in the first four test cases, where the original failure rate was divided between the permanent and "effective transient" fault categories, as the transient failure rates decrease and thus the permanent failure rates increase, the total unreliability of the system increases - as would be expected. In the final test case the transient failure rate was increased while the permanent failure rate was held constant (equal to the original permanent failure rate), which further increased the total system unreliability, as would be expected, due to the added transients.

Test Case	RLMSUB (1,2)	RLMSUB (2,2)	RLM (1,2)	RLM (2,2)	Ratio to Original RLM (1,2)
1.	1.670E-6,	3.000E-7	3.320E-5,	6.000E-6	= 1/3
2.	1.250E-6,	2.250E-7	5.000E-5,	9.000E-6	= 1/2
3.	8.330E-7,	1.500E-7	6.668E-5,	1.200E-5	= 2/3
4.	0.0 ,	0.0	1.000E-4,	1.800E-5	= 1
5.	2.500E-6,	4.500E-7	1.000E-4,	1.800E-5	= 1

Table 7-1 - Failure Rates for FTMP Test Cases

Test Case	Unreliability at 100 min.
1.	0.8866551E-8
2.	0.9490994E-8
3.	1.0085668E-8
4.	1.1187475E-8
5.	1.6950381E-8

Table 7-2 - FTMP Test Case Results at 100 min.

8.0 DESCRIPTION OF MAJOR CARE III DATA STRUCTURES

This section describes the major data structures and memory management schemes used in CARE III. The contents of the coverage plotting files is described along with the major memory management scheme used in module COVRGE. This scheme controls the memory requirements for the numerous single and double-fault function arrays contained in the general fault-handling model. The contents of the reliability plotting file is also described along with the major memory management scheme used in module CARE3. This scheme controls the memory requirements for the numerous functions computed by convolving the coverage-model functions with the reliability-model function.

8.1 COVERAGE PLOTTING FILES DATA STRUCTURE

Plot file SNGFL is created in program COVRGE if the general fault-handling model is chosen - by setting $FLTTYP NAMELIST input parameter 'MARKOV = 2', and if plotting is requested - by setting input parameter 'CVPLOT = .TRUE.'.

File SNGFL contains 9326 word blocks. There are NFTYPS ($FLTTYP NAMELIST input parameter) blocks contained in the file for a maximum of five blocks. Each block consists of the following functions - in the order listed below:

1. $P_B(t)$ - probability single fault is benign
2. $P_{\bar{B}}(t)$ - probability single fault is not benign
3. $p_F(t)$ - single fault failure intensity
4. $P_L(t)$ - probability of latent single fault
5. $p_{DP}(t)$ - single fault detected as permanent intensity.

In program CVGPLT, each block is read into the following COMMON block:

```
 COMMON /SNGFNC/  PBNG(1800) ,PBGSTP ,NPBGST(64)
.                ,PNBNG(1800),PNBSTP ,NPNBST(64)
.                ,PFLD(1800) ,PFSTEP ,NPFSTP(64)
.                ,PLAT(1800) ,PLTSTP ,NPLTST(64)
.                ,PDP(1800)  ,PDPSTP ,NPDPST(64) ,LNORLG
```

The ith block is the ith fault type ITYP defined in program CAREIN. ITYP is used to label the fault type on the plots.

A time array TMAR(1800) is generated prior to each function being plotted using the initial step size variable and step size doubling and halving array corresponding to the function currently being plotted. Function $P_B(t)$ will be used to describe the single-fault functions' plotting data structure, and the manner in which the time array is generated. Each single-fault function is calculated in module COVRGE and stored using the same type of structure.

$P_B(t)$ is stored in array PBNG(IT) using a maximum of 1800 values. Its initial step size is stored in real variable PBGSTP and its step size doubling and halving information is stored in integer array NPBGST(ISTC) using a maximum of 64 step size changes.

The step size may change only by doubling or halving. NPBGST(ISTC) defines how many of each doubled (as a positive integer) or halved (as a negative integer) steps exist per each step size change. If PBGSTP equals 0.2 and 'NPBGST(1 - 7) = 3, 2, 4, -2, -4, 3, 2', with the remainder of the array filled with zeros, then the time array would be computed in program CVGPLT as follows:

```
TMAR( 1) = 0.0 ┐
TMAR( 2) = 0.2 │   initial step size = 0.2
TMAR( 3) = 0.4 ┘
TMAR( 4) = 0.6 ┐
TMAR( 5) = 1.0 │   doubled step size = 0.4
TMAR( 6) = 1.4 ┘
TMAR( 7) = 2.2 ┐
TMAR( 8) = 3.0 │   doubled step size = 0.8
TMAR( 9) = 3.8 │
TMAR(10) = 4.6 ┘
TMAR(11) = 5.0 ┐   halved step size  = 0.4
TMAR(12) = 5.4 ┘
TMAR(13) = 5.6 ┐
TMAR(14) = 5.8 │   halved step size  = 0.2
TMAR(15) = 6.0 ┘
TMAR(16) = 6.2 ┐
TMAR(17) = 6.6 │   doubled step size = 0.4
TMAR(18) = 7.0 │
TMAR(19) = 7.4 ┘
TMAR(20) = 8.2 ┐   doubled step size = 0.8
TMAR(21) = 9.0 ┘
```

Therefore function PBNG(IT), in this truncated example, would consist of 21 points to use for the plot, with index IT ranging from 1 to 21.

Input parameter DBLDF (contained in $FLTTYP NAMELIST) determines the amount of points generated when the single-fault functions are computed in program COVRGE - the smaller the value given to DBLDF the more points generated. Each single-fault function has a unique initial step size and step size change description. That

is why the time array for each function is not contained in the plot file SNGFL. The plot file can contain a maximum of 46,630 words of plotting data, if five fault types were defined by the user. If the time arrays were included in the file, instead of the step size change information, the file would increase to a maximum of 90,005 words of plotting data. This would practically double its size and I/O access time unnecessarily.

Variable LNORLG, the last word contained in each block of plotting data, contains the $FLTTYP NAMELIST input parameter IAXSCV defining the Y-axis scale desired for plotting the fault-handling functions. Program CVGPLT also uses file COVIN, generated by program CAREIN, to retrieve the system tree title to help identify the current run, and the TBASE variable specifying the time base of the plots: 'HRS ', 'MINS', 'SECS' or 'MSEC'.

File DBLFL contains 1865 word blocks. There are NFTYPS squared blocks contained in the file - for a maximum of 25 blocks. Each block consists of the following function:

$p_{DF}(t)$ - double fault failure intensity.

In program CVGPLT, each block is read into the following COMMON block:

```
COMMON /DBLFNC/ PDFAR(1800) ,PDFSTP ,NPDFST(64)
```

The ith block is the ith double-fault type pair (ITYP,JTYP), where JTYP varies the fastest. For example, if three fault types had been defined in program CAREIN, the double-fault failure intensity func-

tion, computed per fault type pair, would be written to DBLFL, in module COVRGE, ordered (1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3).

A time array TMAR(1800) is generated prior to each double-fault function being plotted using the initial step size variable PDFSTP and the step size change description NPDFST(64). The time array is generated in the exact same manner as described above for the single-fault functions.

The plot file DBLFL can contain a maximum of 46,625 words of plotting data, if five fault types were defined. In general there are fewer double-fault functions to plot than single-fault functions because there are five times NFTYPS single-fault functions and NFTYPS squared double-fault functions, in files SNGFL and DBLFL respectively. At its maximum size, it would contain 90,000 words of plotting data if the time arrays were passed in the plotting file. DBLFL does not contain the IAXSCV input parameter. Program CVGPLT uses variable LNORLG from file SNGFL for the choice of the Y-axis scale.

8.2 RELIABILITY PLOTTING FILE DATA STRUCTURE

Plot file PLTFL is created in program CARE3, if input parameter RLPLOT is set .TRUE. in $STAGES NAMELIST. File PLTFL contains 260 words of plotting data that consists of one 195 word block and one 65 word block. The first block contains the CARE3 summary results: "Q SUM", "P* SUM", and "Q+P* SUM" read by program RELPLT into COMMON

/PLTCOM/ QLTSUM(65) ,PSTSUM(65) ,QPSTSM(65). The second block contains the time values corresponding to the summary functions' results and is read into array TMAR(65) contained in the following COMMON block:

```
      COMMON /STEPCM/  ITSTPS   , MAXSTP   , RELSTP   , TBASE
     .               , TMAR(65) , NSTGRN   , KWT      , PSTRNC
     .               , PRCODE   , RLPLOT   , IAXSRL   , QPTRNC
     .               , CPLFLG   , SYSMNT   , TBCF
C
      CHARACTER*4      PRCODE
C
      LOGICAL          RLPLOT   , SYSMNT   , CPLFLG
```

Due to the unequal step sizes used in computing the enhanced CARE3 functions, if input parameter LGTMST was set .TRUE. in $FLTTYP NAMELIST, the functions' corresponding time values are written to file PLTFL, and hence no longer generated in program RELPLT. This increases the size of file PLTFL by only 65 words since all three functions were computed using the same step sizes.

Program RELPLT also uses file RELIN, generated by program CAREIN, to retrieve the system tree title to help identify the current run, the TBASE variable specifying the time base of the plots, and input parameter IAXSRL to determine the Y-axis plotting choice.

8.3 COMMON BLOCK BXYCOM DATA STRUCTURE

The purpose of this section is to detail the data structure and memory management scheme used for the arrays in COMMON /BXYCOM/ contained in module CARE3. Therefore the way the data is handled rather than what the data is will be described in detail. (See refs. 5, 6 and 8 for detailed data descriptions.)

Table 8-1 is a partial listing of COMMON block BXYCOM including a brief description of the arrays and the contents of the COMMON block. The equivalenced arrays are not shown here but are listed in Tables 8-2, 8-3, 8-4, 8-5 and 8-6 that follow the COMMON block listing.

The tables' format is structured so that all pertinent array equivalences are shown in pairs. The pairs consist of the name and index of the first word of the array followed by the name and index of the last word of the array. The variables used for the indices are also listed in the tables along with the segment number 'SEG'. This number is used to manage the data in main memory and when buffering is required. SUBROUTINE BUFDAT manages all data indexed using a segment number and determines whether buffering to disk is required.

For the tables that contain segment numbers: Tables 8-2, 8-5 and 8-6, three levels of array equivalences are shown. The array listed in the first column is the array contained in the COMMON block statement - either NXYAR or BXYAR; the array listed in the second column is the equivalenced array used to buffer the data using SUBROUTINE BUFDAT; and the array(s) listed in the third column are the equivalenced array(s) used in the code when generating and accessing the data. Tables 8-3 and 8-4 contain arrays that are never buffered to disk while module CARE3 is executing. The first column of these tables contains the COMMON block array name NXYAR or BXYAR equivalenced to array(s) used in the code when generating and accessing the data. If desired, see module CARE3 for the DIMENSION and EQUIVALENCE code statements that define the arrays listed in these tables.

```
C*****************************************************************************
C*                                                                           *
C*              COMMON BLOCK : BXYCOM                                        *
C*                                                                           *
C*              NXYAR     =  BUFFER ARRAY FOR NXX AND NXY DATA               *
C*                                                                           *
C*              BFNXX     =  BUFFER FOR NXX DATA                             *
C*              BFNXY     =  BUFFER FOR NXY DATA                             *
C*                                                                           *
C*              HLTP      =  1. / SUM( HL(XI.P,T))                           *
C*              HNBP      =  SUM( HNB(XI.P,T))                               *
C*              HNBT      =  SUM( HNB(XI.T,T))                               *
C*              HBNG      =  HB(X,I,T)                                       *
C*              HBNGSB    =  HB(X,I.T,T) FOR STAGES WITH INTERNALLY          *
C*                           REDUNDANT MODULES                               *
C*              HDPTIN    =  INTEGRATED HDP(X,I.T,T)                         *
C*              FLMP      =  SUM( LAM(XI.P,T))                               *
C*              FLMT      =  SUM( LAM(XI.T,T))                               *
C*                                                                           *
C*              RXAR      =  R(X:T)                                          *
C*              CMPRAR    =  1.0 COMPLEMENT OF 'RXAR'                        *
C*              SMCLAM    =  SUM OF 'CLAM'S                                  *
C*              AXAR      =  A(X:T)                                          *
C*              SRPSTF    =  SUBRUN P* FUNCTION (T:X)                        *
C*              FPMX      =  P(MUX,T:LX)                                     *
C*              PSLX      =  P*(T:L-1X) * (NX-LX+1)                          *
C*              IJSGIN    =  NUMBER ASSIGNED TO A PAIR OF C.P. STAGES        *
C*                                                                           *
C*              BXYAR     =  BUFFER ARRAY FOR BXX AND BXY DATA               *
C*                                                                           *
C*              BFBXX     =  BUFFER FOR BXX DATA                             *
C*              BFBXY     =  BUFFER FOR BXY DATA                             *
C*                                                                           *
C*              IQXNOP    =  NUMBER OF 'IN-USE' MODULES                      *
C*              KQXNOP    =   INDEX OF 'IN-USE' MODULES                      *
C*              KFSTG     =  NUMBER OF THE FIRST STAGE IN A SUBRUN           *
C*                                                                           *
C*              KNT       =  C.P. COUNT FOR A SUBRUN                         *
C*              ICSTG     =  NUMBER OF COUPLED STAGES IN A SUBRUN            *
C*              NCSTG     =  NUMBER OF STAGES IN A SUBRUN                    *
C*              IISTG     =  POINTER INTO KNT ARRAY                          *
C*              IUSTG     =  POINTER INTO KNT ARRAY                          *
C*                                                                           *
C*              NXX       =  N DATA FOR XX CASE                              *
C*              NINXX     =   INDEX OF CURRENT 'SEG' FOR NXX DATA            *
C*              NBNXX     =  NUMBER OF CURRENT BLOCK FOR NXX DATA            *
C*              NSNXX     =  NUMBER OF 'SEG' / BLOCK FOR NXX DATA            *
C*              NWNXX     =  NUMBER OF WORDS / BLOCK FOR NXX DATA            *
C*                                                                           *
```

(COMMON block BXYCOM continued on following page)

```
C*              NXY       =  N DATA FOR XY CASE                          *
C*              NINXY     =    INDEX OF CURRENT 'SEG' FOR NXY DATA       *
C*              NBNXY     =  NUMBER OF CURRENT BLOCK FOR NXY DATA        *
C*              NSNXY     =  NUMBER OF 'SEG' / BLOCK FOR NXY DATA        *
C*              NWNXY     =  NUMBER OF WORDS / BLOCK FOR NXY DATA        *
C*                                                                       *
C*              BXX       =  B DATA FOR XX CASE                          *
C*              XXDFP     = 8-4 TERM FOR SUMA - P,X,X                    *
C*              XXDFT     = 8-4 TERM FOR SUMA - T,X,X                    *
C*              XXNBP1    = 8-5 TERM FOR SUMC - P,X,X                    *
C*              XXNBT1    = 8-5 TERM FOR SUMC - T,X,X                    *
C*              XXNBP2    = 8-5 TERM FOR SUMA - P,X,X                    *
C*              XXNBT2    = 8-5 TERM FOR SUMA - T,X,X                    *
C*              XXFLP     =  A' TERM FOR SUMA - P,X,X                    *
C*              XXFLT     =  A' TERM FOR SUMA - T,X,X                    *
C*              NIBXX     =    INDEX OF CURRENT 'SEG' FOR BXX DATA       *
C*              NBBXX     =  NUMBER OF CURRENT BLOCK FOR BXX DATA        *
C*              NSBXX     =  NUMBER OF 'SEG' / BLOCK FOR BXX DATA        *
C*              NWBXX     =  NUMBER OF WORDS / BLOCK FOR BXX DATA        *
C*                                                                       *
C*              BXY       =  B DATA FOR XY CASE                          *
C*              XYDFP     = 8-4 TERM FOR SUMA - P,X,Y                    *
C*              YXDFP     = 8-4 TERM FOR SUMA - P,Y,X                    *
C*              XYDFT     = 8-4 TERM FOR SUMA - T,X,Y                    *
C*              YXDFT     = 8-4 TERM FOR SUMA - T,Y,X                    *
C*              XYNBP1    = 8-5 TERM FOR SUMC - P,X,Y                    *
C*              YXNBP1    = 8-5 TERM FOR SUMC - P,Y,X                    *
C*              XYNBT1    = 8-5 TERM FOR SUMC - T,X,Y                    *
C*              YXNBT1    = 8-5 TERM FOR SUMC - T,Y,X                    *
C*              XYNBP2    = 8-5 TERM FOR SUMA - P,X,Y                    *
C*              YXNBP2    = 8-5 TERM FOR SUMA - P,Y,X                    *
C*              XYNBT2    = 8-5 TERM FOR SUMA - T,X,Y                    *
C*              YXNBT2    = 8-5 TERM FOR SUMA - T,Y,X                    *
C*              NIBXY     =    INDEX OF CURRENT 'SEG' FOR BXY DATA       *
C*              NBBXY     =  NUMBER OF CURRENT BLOCK FOR BXY DATA        *
C*              NSBXY     =  NUMBER OF 'SEG' / BLOCK FOR BXY DATA        *
C*              NWBXY     =  NUMBER OF WORDS / BLOCK FOR BXY DATA        *
C*                                                                       *
C*              INXY      =  UNIT NUMBER OF NXX AND NXY DATA             *
C*              IBXY      =  UNIT NUMBER OF BXX AND BXY DATA             *
C*                                                                       *
C************************************************************************
C*                                                                       *
      COMMON /BXYCOM / INXY , IBXY
     .               , NIBXX, NBBXX, NSBXX, NWBXX
     .               , NIBXY, NBBXY, NSBXY, NWBXY
     .               , NINXX, NBNXX, NSNXX, NWNXX
     .               , NINXY, NBNXY, NSNXY, NWNXY
     .               , NXYAR(30830), BXYAR(33001)
C*                                                                       *
C************************************************************************
```

Table 8-1 - Description of BXYCOM Variables

ARRAY EQUIVALENCES				INDEXED BY 'SEG'	
NXYAR(1) = BFNXX	(1, 1) = NXX	(1, 1)	(KQX,NINXX)	1	
NXYAR(10) = BFNXX	(10, 1) = NXX	(10, 1)			
NXYAR(11) = BFNXX	(1, 2) = NXX	(1, 2)		2	
NXYAR(20) = BFNXX	(10, 2) = NXX	(10, 2)			
.	.	.			
.	.	.		3 - 19	
.	.	.			
NXYAR(191) = BFNXX	(1,20) = NXX	(1,20)		20 = NSNXX	
NXYAR(200) = BFNXX	(10,20) = NXX	(10,20)			
NXYAR(201) = BFNXY	(1, 1) = NXY	(1, 1)	(KQXY,NINXY)	1	
NXYAR(300) = BFNXY	(100, 1) = NXY	(100, 1)			
NXYAR(301) = BFNXY	(1, 2) = NXY	(1, 2)		2	
NXYAR(400) = BFNXY	(100, 2) = NXY	(100, 2)			
.	.	.			
.	.	.		3 - 49	
.	.	.			
NXYAR(5101) = BFNXY	(1,50) = NXY	(1,50)		50 = NSNXY	
NXYAR(5200) = BFNXY	(100,50) = NXY	(100,50)			

Table 8-2 - N-XY Critical Pair Counts Per Subrun

ARRAY EQUIVALENCES					INDEXED BY
NXYAR(5201) = HLTP	(1, 1)				(ISTG,IT)
NXYAR(6500) = HLTP	(20,65)				
NXYAR(6501) = HNBP	(1, 1)				(ISTG,IT)
NXYAR(7800) = HNBP	(20,65)				
NXYAR(7801) = HNBT	(1, 1)				(ISTG,IT)
NXYAR(9100) = HNBT	(20,65)				
NXYAR(9101) = HBNG	(1,1, 1)	= HDPTIN	(1,1, 1)		(ISTG,ICAT,IT)
NXYAR(15600) = HBNG	(20,5,65)	= HDPTIN	(20,5,65)		
NXYAR(15601) = FLMP	(1,1)				(ISTG,IT)
NXYAR(16900) = FLMP	(1,1)				
NXYAR(16901) = FLMT	(1, 1)				(ISTG,IT)
NXYAR(18200) = FLMT	(20,65)				
NXYAR(18201) = RXAR	(1, 1)				(ISTG,IT)
NXYAR(19500) = RXAR	(20,65)				
NXYAR(19501) = CMPRAR	(1, 1)	= SMCLAM	(1, 1)		(ISTG,IT)
NXYAR(20800) = CMPRAR	(20,65)	= SMCLAM	(20,65)		
NXYAR(20801) = AXAR	(1, 1)	= SRPSTF	(1, 1)		(ISTG,IT),
NXYAR(22100) = AXAR	(20,65)	= SRPSTF	(65,20)		(IT,ISTG)
NXYAR(22101) = FPMX	(1, 1, 1)				(ISTG,IMUX,IPMX) or
NXYAR(24100) = FPMX	(20,10,10)				(JSTG,IMUY,JPMY)
NXYAR(24101) = PSLX	(1)				(ISTG)
NXYAR(24120) = PSLX	(20)				
NXYAR(24121) = IJSGIN	(1)				(IJSTG)
NXYAR(24330) = IJSGIN	(210)				
NXYAR(24331) = HBNGSB	(1,1, 1)				(ISTG,ICAT,IT)
NXYAR(30830) = HBNGSB	(20,1,65)				

Table 8-3 - (ISTG) Single-Fault Reliability Functions Per Subrun

ARRAY EQUIVALENCES			INDEXED BY
BXYAR(1)	= IQXNOP(1, 1)		(KQX,ISTG) or
BXYAR(200)	= IQXNOP(10,20)		(KQY,JSTG)
BXYAR(201)	= KQXNOP(1, 1)		(IMUX,ISTG)
BXYAR(400)	= KQXNOP(10,20)		(IMUY,JSTG)
BXYAR(401)	= KFSTG		
BXYAR(402)	= KNT(1, 1, 1)		(IIDX,ISTG,KQX) or
BXYAR(14401)	= KNT(70,20,10)		(IIDX,JSTG,KQY)
BXYAR(14402)	= ICSTG		(Not Used in CARE3)
BXYAR(14403)	= NCSTG		(Not Used in CARE3)
BXYAR(14404)	= IISTG	(1)	(Not Used
BXYAR(14473)	= IISTG	(70)	in CARE3)
BXYAR(14474)	= IUSTG	(1)	(ISTG) or
BXYAR(14543)	= IUSTG	(70)	(ISTG+1)
.			(Unused Portion of
.			BXYAR Buffered to
BXYAR(15000)			and from Unit 14)

Table 8-4 - Portion of BXYAR Generated in CAREIN Module Per Subrun

ARRAY EQUIVALENCES			INDEXED BY 'SEG'
BXYAR(402) =	BFBXX(1, 1) = BXX (1, 1)	(IMUX,NIBXX)	1
	BFBXX(10, 1) = BXX (10, 1)		
	BFBXX(11, 1) = XXDFP (1, 1)	(IT,NIBXX)	1
	BFBXX(75, 1) = XXDFP (65, 1)		
	BFBXX(76, 1) = XXDFT (1, 1)	(IT,NIBXX)	1
	BFBXX(140, 1) = XXDFT (65, 1)		
	BFBXX(141, 1) = XXNBP1(1, 1)	(IT,NIBXX)	1
	BFBXX(205, 1) = XXNBP1(65, 1)		
	BFBXX(206, 1) = XXNBT1(1, 1)	(IT,NIBXX)	1
	BFBXX(270, 1) = XXNBT1(65, 1)		
	BFBXX(271, 1) = XXNBP2(1, 1)	(IT,NIBXX)	1
	BFBXX(335, 1) = XXNBP2(65, 1)		
	BFBXX(336, 1) = XXNBT2(1, 1)	(IT,NIBXX)	1
	BFBXX(400, 1) = XXNBT2(65, 1)		
	BFBXX(401, 1) = XXFLP (1, 1)	(IT,NIBXX)	1
	BFBXX(465, 1) = XXFLP (65, 1)		
	BFBXX(466, 1) = XXFLT (1, 1)	(IT,NIBXX)	1
BXYAR(931) =	BFBXX(530, 1) = XXFLT (65, 1)		
BXYAR(932) =	BFBXX(1, 2) = BXX (1, 2)	(IMUX,NIBXX)	2
	BFBXX(10, 2) = BXX (10, 2)		
	. .		
	. .		
	BFBXX(466, 2) = XXFLT (1, 2)	(IT,NIBXX)	2
BXYAR(1461) =	BFBXX(530, 2) = XXFLT (65, 2)		
	. .		
	. .		3 - 19
	. .		
BXYAR(10472) =	BFBXX(1,20) = BXX (1,20)	(IMUX,NIBXX)	20
	BFBXX(10,20) = BXX (10,20)		
	. .		
	. .		
	BFBXX(466,20) = XXFLT (1,20)	(IT,NIBXX)	20 = NSBXX
BXYAR(11001) =	BFBXX(530,20) = XXFLT (65,20)		

Table 8-5 - (ISTG,ISTG) Double-Fault Reliability Functions Per Subrun

ARRAY EQUIVALENCES			INDEXED BY 'SEG'
BXYAR(11002) =	BFBXY(1, 1) = BXY (1, 1)	(KMXY,NIBXY)	1
	BFBXY(100, 1) = BXY (100, 1)		
	BFBXY(101, 1) = XYDFP (1, 1)	(IT,NIBXY)	1
	BFBXY(165, 1) = XYDFP (65, 1)		
	BFBXY(166, 1) = YXDFP (1, 1)	(IT,NIBXY)	1
	BFBXY(230, 1) = YXDFP (65, 1)		
	BFBXY(231, 1) = YXDFT (1, 1)	(IT,NIBXY)	1
	BFBXY(360, 1) = YXDFT (65, 1)		
	BFBXY(361, 1) = XYNBP1(1, 1)	(IT,NIBXY)	1
	BFBXY(425, 1) = XYNBP1(65, 1)		
	BFBXY(426, 1) = YXNBP1(1, 1)	(IT,NIBXY)	1
	BFBXY(490, 1) = YXNBP1(65, 1)		
	BFBXY(491, 1) = XYNBT1(1, 1)	(IT,NIBXY)	1
	BFBXY(555, 1) = XYNBT1(65, 1)		
	BFBXY(556, 1) = YXNBT1(1, 1)	(IT,NIBXY)	1
	BFBXY(620, 1) = YXNBT1(65, 1)		
	BFBXY(621, 1) = XYNBP2(1, 1)	(IT,NIBXY)	1
	BFBXY(685, 1) = XYNBP2(65, 1)		
	BFBXY(686, 1) = YXNBP2(1, 1)	(IT,NIBXY)	1
	BFBXY(750, 1) = YXNBP2(65, 1)		
	BFBXY(751, 1) = XYNBT2(1, 1)	(IT,NIBXY)	1
	BFBXY(815, 1) = XYNBT2(65, 1)		
	BFBXY(816, 1) = YXNBT2(1, 1)	(IT,NIBXY)	1
BXYAR(11881) =	BFBXY(880, 1) = YXNBT2(65, 1)		
BXYAR(11882) =	BFBXY(1, 2) = BXY (1, 2)	(KMXY,NIBXY)	2
	BFBXY(100, 2) = BXY (100, 2)		
	. .		
	. .		
	BFBXY(816, 2) = YXNBT2(1, 2)	(IT,NIBXY)	2
BXYAR(12761) =	BFBXY(880, 2) = YXNBT2(65, 2)		
	. .		
	. .		3 - 24
	. .		
BXYAR(32122) =	BFBXY(1,25) = BXY (1,25)	(KMXY,NIBXY)	25
	BFBXY(100,25) = BXY (100,25)		
	. .		
	. .		
	BFBXY(816,25) = YXNBT2(1,25)	(IT,NIBXY)	25 = NSBXY
BXYAR(33001) =	BFBXY(880,25) = YXNBT2(65,25)		

Table 8-6 - (ISTG,JSTG) Double-Fault Reliability Functions Per Subrun

COMMON /BXYCOM/ contains two large buffer arrays and several data controlling variables and constants described below:

1) NXYAR(30830) is the buffer array containing the N-XY critical-pair counts (see Table 8-2), and the single-fault reliability function arrays (see Table 8-3);

2) BXYAR(33001) is the buffer array containing the B-XY data generated in the CAREIN module and used in the CARE3 module (see Table 8-4), the B-XY probabilities that critical pairs exist between specific modules in stage x and stage y, and the double-fault reliability functions (see Tables 8-5 and 8-6);

3) INXY is the unit number (= 18) of the disk file used to hold the buffer BFNXY array (see Table 8-2) containing the NXY critical-pair counts, if the number of critical-pair stage combinations (ISTG,JSTG) exceed constant NSNXY (= 50) per subrun;

4) IBXY is the unit number (= 19) of the disk file used to hold the buffer BFBXY array (see Table 8-6), if the number of (ISTG,JSTG) double-fault pairs of stages exceeds constant NSBXY (= 25) per subrun;

5) NSNXX is the constant (= 20) defining the number of segments 'SEG' per block used when generating the buffer BFNXX array (see Table 8-2);

6) NWNXX is the constant (= 200) defining the number of words per block used when generating the buffer BFNXX array (see Table 8-2);

7) NSNXY is the constant (= 50) defining the number of segments 'SEG' per block used when generating and buffering the BFNXY array (see Table 8-2);

8) NWNXY is the constant (= 5000) defining the number of words per block used when generating and buffering the BFNXY array (see Table 8-2);

9) NSBXX is the constant (= 20) defining the number of segments 'SEG' per block used when generating the buffer BFBXX array (see Table 8-5);

10) NWBXX is the constant (= 10600) defining the number of words per block used when generating the buffer BFBXX array (see Table 8-5);

11) NSBXY is the constant (= 25) defining the number of segments 'SEG' per block used when generating and buffering the BFBXY array (see Table 8-6);

12) NWBXY is the constant (= 22000) defining the number of words per block used when generating and buffering the BFBXY array (see Table 8-6);

13) NINXX is the index variable of the current 'SEG' of the NXX data; it ranges from 1 to NSNXX (see Table 8-2);

14) NBNXX is the number of the current block for the NXX data; no coded upper limit exists for NBNXX;

15) NINXY is the index variable of the current 'SEG' of the NXY data; it ranges from 1 to NSNXY (see Table 8-2);

16) NBNXY is the number of the current block for the NXY data; no coded upper limit exists for NBNXY;

17) NIBXX is the index variable of the current 'SEG' of the BXX data; it ranges from 1 to NSBXX (see Table 8-5);

18) NBBXX is the number of the current block for the BXX data; no coded upper limit exists for NBBXX;

19) NIBXY is the index variable of the current 'SEG' of the BXY data; it ranges from 1 to NSBXY (see Table 8-6);

20) NBBXY is the number of the current block for the BXY data; no coded upper limit exists for NBBXY.

Tables 8-2 and 8-3 show that NXYAR is partitioned into three sections. The first 200 (NWNXX) words of NXYAR contain the intra-stage (stages with modules that are critically coupled to other modules in the same stage) critical-pair counts. A maximum of 20 (NSNXX) such stage pairings or segments, indexed by NINXX, can be stored in main memory per subrun. Since the maximum number of stages per subrun is 20, and thus the maximum number of pairings is 20, no buffering is required of the NXX array.

The next 5000 (NWNXY) words of NXYAR contain the inter-stage (stages with modules that are critically coupled to modules in a different stage) critical-pair counts. A maximum of 50 (NSNXY) such

stage pairings or segments, indexed by NINXY, can by stored in main memory per subrun before buffering to disk is required. When the number of separate critically coupled stages in a given subrun exceeds 10, buffering to disk of the NXY data will be necessary. The number of (ISTG,JSTG) pairs, with ISTG less than JSTG, equals 55 for 11 critically coupled stages and increases to 190 for 20 critically coupled stages (see Table 8-7 below). Thus a maximum of four blocks would be written initially to unit INXY, for a given subrun containing 20 critically coupled stages, and read per fault vector computation.

Number of Critically Coupled Stages	Number of (ISTG,JSTG) pairs where ISTG < JSTG
2	1
3	3
4	6
5	10
6	15
7	21
8	28
9	36
10	45
11	55
12	66
13	78
14	91
15	105
16	120
17	136
18	153
19	171
20	190

Table 8-7 - Number of Stage Pairs Given Number of Critically Coupled Stages

The remaining 25,630 words of NXYAR contain the single-fault reliability functions per subrun. These are the functions that are generated using the single-fault coverage moment functions, passed from module COVRGE, convolved with the reliability-model function. No buffering to disk is required of the arrays contained in this portion of NXYAR.

Tables 8-4, 8-5 and 8-6 show that BXYAR is also partitioned into three sections. These partitions overlap because the data is required at different times during the execution of CARE3. The first 15,000 words of BXYAR contain the critical-pair data per subrun generated in module CAREIN. This data is used to generate the NXX and NXY arrays contained in NXYAR. Once these arrays are computed, BXYAR(402) through (15000) can be used for the double-fault reliability functions (see Tables 8-5 and 8-6). Notice that words BXYAR (14402) through (14473) and BXYAR(14544) through (15000) do not contain data used in module CARE3, while the later range of buffered words are not used in any module. If array IUSTG(70) were moved directly below array KNT(70,20,10), in COMMON /BXYCOM/ in both modules CAREIN and CARE3, and only BXYAR(1) through (14471) were buffered to unit 14, 529 less words would have to be buffered to and from disk per subrun containing critically coupled stages.

The second partition of BXYAR consists of words BXYAR(402) through (11001) and contains the BXX function array and the intra-stage double-fault reliability functions. These are the functions

computed using the double-fault coverage moment functions convolved with the reliability-model function. These function arrays begin with the letters XX (see Table 8-5). A maximum of 20 (NSBXX) such stage pairings or segments, indexed by NIBXX, can be stored in main memory per subrun. No buffering to disk is ever required of this data since 20 pairings is the maximum number possible per subrun.

The double-fault reliability function arrays that begin with the letters XX are dimensioned (530,1) yet are indexed by IT (time step: 1 to 65) and NIBXX (segment number: 1 to 20). The (530,1) DIMENSION statement is necessary since there are 530 words contained in each segment of this data type. For example, array XXDFP(1-65,3) is two segments from the beginning of XXDFP(1-65,1), i.e. XXDFP(1,3) is (2 * 530) words offset from XXDFP(1,1).

The third partition of BXYAR consists of words BXYAR(11002) through (33001) and contains the BXY function array and the inter-stage double-fault reliability functions. A maximum of 25 (NSBXY) such stage pairings or segments, indexed by NIBXY, can be stored in main memory per subrun before buffering to disk is required. When the number of separate critically coupled stages in a given subrun exceeds seven, buffering to disk of the BFBXY buffer (see Table 8-6) will be necessary. The number of (ISTG,JSTG) pairs, with ISTG less than JSTG, equals 28 for eight critically coupled stages and increases to 190 for 20 critically coupled stages (see Table 8-7). Thus a maximum of eight blocks would be written initially to unit IBXY, for a

given subrun containing 20 critically coupled stages, and read per fault vector computation.

The inter-stage double-fault reliability function arrays, contained in this partition, are dimensioned (880,1) because there are 880 words contained per segment for this data type. Given an (ISTG, JSTG) pair, with ISTG less than JSTG, the arrays that begin with the letters XY are created using the stage pair ordering (JSTG,ISTG); the arrays that begin with the letters YX are created using the stage pair ordering (ISTG,JSTG).

When modeling a large system, if separate critical pair trees are defined so that each critical-pair tree contains a maximum of seven critically coupled stages (see ref. 2), no buffering to and from disk will be necessary for any of the functions described above. This will enable the program to execute as fast as is possible.

9.0 ADDITIONAL TESTING OF USER INPUT VALUES

Several additional warning/error checks and messages were added to SUBROUTINE VLDNML in module CAREIN to aid the user in defining a valid input stream. The additional checks deal with the following six areas:

1) Ratio of ALP(i) to BET(i) should be less than 1000 to avoid COVRGE numerical instability problems. - WARNING

2) Reciprocal of failure rates λ to the power ω should be at least three orders of magnitude greater than the transition times in the fault-handling model. - WARNING

3) Coverage plot files SNGFL and DBLFL can only be generated using the general coverage model (MARKOV = 2). - WARNING

4) Runs with mission times of one hour or more (FT >= 1 hour) should use the logarithmic time step method for all function computations (LGTMST = .TRUE.) for greater accuracy without increasing execution time. - WARNING

5) A DEL(i) (or RHO(i)) value of zero is illegal if the DEL(i) (or RHO(i)) parameter is specified as a uniform rate function: IDELF(i) = 2 (or IRHOF(i) = 2). - ERROR

6) Permanent fault types only must be assigned to stage(s) with internally redundant modules. The fault type(s) assigned to the redundant submodules, using JSBTYP(i,x), and the fault type(s) assigned to the non-redundant portion of the module or the redundant portion where no redundant submodules remain, using JTYP(i,x), must have ALP(i) = 0.0. - ERROR

Valid range checks were also added to SUBROUTINE VLDNML for all new input parameters added during this enhancement phase.

The modification to the FTMP test case, contained in figure 9-1, was one of the test cases used to check the added error tests. The output generated by CAREIN using the error filled CREIN input file is shown in figure 9-2 .

```
3,     10.0,     1.0E+6,        0.0,
        1.0,        1.0,        0.0,
       10.0,       36.0,       10.0,
        0.0,        0.0,        0.0,
        0.0,        0.0,        0.0,
          1,          1,          1,
          1,          1,          1,
          1,          1,          1,
        1.0,        1.0,        1.0,
        1.0,        1.0,        0.0,
        1.0,        1.0,        1.0,
       0.05,     1.0E-5,    .TRUE., .TRUE., , 1, .FALSE./
3,       15,          9,          5,
         11,          5,          3,
          0,         44,          0,
          0,         40,          0,
     .TRUE.,    .FALSE.,    .TRUE.,
           ,  ,  ,  ,  ,
           ,  ,  ,  ,  ,
          3,  ,  ,  ,  ,
           ,  ,  ,  4/
2,  2,  1,
 ,   ,
3,  2,
2,
 ,   ,
3,  3,
 ,
 ,   ,
 ,   ,
 ,
1.0E-1, 0.18E-1,
1.0E-4, 0.18E-4,
1.0E-6,
 ,   ,
 ,   ,
 ,
 ,   ,
1.0E-1, 0.18E-1, /
100.0,  50,  2,  .TRUE.,  .TRUE.,  ,  1.0E-10/
FTMP ARCHITECTURE - 15 PROCESSORS, 9 MEMORY MODULES, 5 BUSES
WITH CRITICAL FAULT PAIRS AND INTERNALLY REDUNDANT MEMORY MODULES.
DECEMBER, 1984 - With (MSUB*RLMSUB) + RLM = ORIGINAL LAMBDA
                 and                      RLM = 2 * ORIGINAL LAMBDA / 3
1 3 4 4
4 0 1 2 3

.
.  (Critical Pair Fault-Tree not listed)
.
```

Figure 9-1 - FTMP Modified Test Case (containing input errors)

```
 CCCCC        A        RRRR      EEEEE      IIIIIIIIIIIIIII
C            A A       R   R     E            I     I     I
C           A   A      RRRR      EEE          I     I     I
C           A AAA A    R   R     E            I     I     I
 CCCCC     A       A   R    R    EEEEE      IIIIIIIIIIIIIII
```

WARNING RATIO OF TRANSITION PARAMETERS ALP = 1.000e+06 AND BET = 1.000e+00, FOR FAULT TYPE 2, EXCEEDS REASONABLE BOUND OF 1000. PROGRAM COVRGE MAY BECOME NUMERICALLY UNSTABLE

WARNING COVERAGE PLOTS REQUESTED WITH THE HOMOGENEOUS MARKOV MODEL. PLOT FILES SNGFL AND DBLFL CAN BE GENERATED ONLY WITH THE GENERAL SOLUTION; SET INPUT PARAMETER MARKOV = 2 AND RERUN

WARNING FAULT OCCURRENCE RATE BASED ON RLM(1, 1) AND OMG(1, 1) EQUALS 1.000e+01 HOURS AND IS NOT AT LEAST 3 ORDERS OF MAGNITUDE LARGER THAN ITS CORRESPONDING FAULT-HANDLING RATE PARAMETER BASED ON ALP(1), WHICH EQUALS 1.000e-01 HOURS. CARE III'S MATHEMATICAL APPROXIMATIONS REQUIRE THIS SEPARATION

WARNING FAULT OCCURRENCE RATE BASED ON RLM(1, 1) AND OMG(1, 1) EQUALS 1.000e+01 HOURS AND IS NOT AT LEAST 3 ORDERS OF MAGNITUDE LARGER THAN ITS CORRESPONDING FAULT-HANDLING RATE PARAMETER BASED ON BET(1), WHICH EQUALS 1.000e+00 HOURS. CARE III'S MATHEMATICAL APPROXIMATIONS REQUIRE THIS SEPARATION

WARNING FAULT OCCURRENCE RATE BASED ON RLM(1, 1) AND OMG(1, 1) EQUALS 1.000e+01 HOURS AND IS NOT AT LEAST 3 ORDERS OF MAGNITUDE LARGER THAN ITS CORRESPONDING FAULT-HANDLING RATE PARAMETER BASED ON DEL(1), WHICH EQUALS 1.000e-01 HOURS. CARE III'S MATHEMATICAL APPROXIMATIONS REQUIRE THIS SEPARATION

WARNING FAULT OCCURRENCE RATE BASED ON RLM(2, 1) AND OMG(2, 1) EQUALS 5.556e+01 HOURS AND IS NOT AT LEAST 3 ORDERS OF MAGNITUDE LARGER THAN ITS CORRESPONDING FAULT-HANDLING RATE PARAMETER BASED ON ALP(1), WHICH EQUALS 1.000e-01 HOURS. CARE III'S MATHEMATICAL APPROXIMATIONS REQUIRE THIS SEPARATION

(CAREIN Output Listing continued on following page)

```
**WARNING**  FAULT OCCURRENCE RATE BASED ON     RLM(2, 1) AND     OMG(2, 1)
             EQUALS  5.556e+01 HOURS AND IS NOT AT LEAST 3 ORDERS OF
             MAGNITUDE LARGER THAN ITS CORRESPONDING FAULT-HANDLING RATE
             PARAMETER BASED ON    BET(1), WHICH EQUALS  1.000e+00 HOURS.
             CARE III'S MATHEMATICAL APPROXIMATIONS REQUIRE THIS SEPARATION

**WARNING**  FAULT OCCURRENCE RATE BASED ON     RLM(2, 1) AND     OMG(2, 1)
             EQUALS  5.556e+01 HOURS AND IS NOT AT LEAST 3 ORDERS OF
             MAGNITUDE LARGER THAN ITS CORRESPONDING FAULT-HANDLING RATE
             PARAMETER BASED ON    DEL(1), WHICH EQUALS  1.000e-01 HOURS.
             CARE III'S MATHEMATICAL APPROXIMATIONS REQUIRE THIS SEPARATION

**WARNING**  FAULT OCCURRENCE RATE BASED ON RLMSUB(1, 2) AND OMGSUB(1, 2)
             EQUALS  1.000e+01 HOURS AND IS NOT AT LEAST 3 ORDERS OF
             MAGNITUDE LARGER THAN ITS CORRESPONDING FAULT-HANDLING RATE
             PARAMETER BASED ON    DEL(3), WHICH EQUALS  1.000e-01 HOURS.
             CARE III'S MATHEMATICAL APPROXIMATIONS REQUIRE THIS SEPARATION

**ERROR**  MUST ASSIGN A PERMANENT FAULT TYPE ('ALP' = 0.0) TO  JTYP(2, 2)
           FOR THIS STAGE WITH INTERNALLY REDUNDANT MODULES

**WARNING**  FAULT OCCURRENCE RATE BASED ON RLMSUB(2, 2) AND OMGSUB(2, 2)
             EQUALS  5.556e+01 HOURS AND IS NOT AT LEAST 3 ORDERS OF
             MAGNITUDE LARGER THAN ITS CORRESPONDING FAULT-HANDLING RATE
             PARAMETER BASED ON    DEL(3), WHICH EQUALS  1.000e-01 HOURS.
             CARE III'S MATHEMATICAL APPROXIMATIONS REQUIRE THIS SEPARATION

**WARNING**  EQUAL TIME STEPS WERE REQUESTED WITH A MISSION TIME GREATER
             THAN 1 HOUR.  CARE III'S CALCULATIONS ARE MORE ACCURATE,
             ESPECIALLY FOR LONG MISSION TIMES, WITH INPUT PARAMETER
             LGTMST SET .TRUE. - WITHOUT INCREASING EXECUTION TIME
```

Figure 9-2 - CAREIN Output Listing (generated from error filled input file)

10.0 CONCLUDING REMARKS

The CARE III computer program has been greatly improved since the original version 3 was released to NASA in 1982. Boeing Computer Services made extensive improvements to the program. BCS concentrated their efforts in the area of the critical pair functions' calculations and the data structures used to manage the numerous function arrays. They also added an efficient exponential coverage model.

During this enhancement phase, Sequoia Systems, Inc. enhanced the computer program extensively in the area of the Kolmogorov forward equation solution and with the addition of an internally redundant modeling capability. CARE III is now accurate over the entire time range from zero to extremely large operating times. And the user may now define stages, as input to CARE III, where modules within a stage are internally redundant. The numerical method used to solve the general coverage model's Volterra equations was slightly improved upon by allowing the adaptive integration step size to halve as well as double.

Several coding and conversion errors were discovered and corrected during this enhancement phase. Some were discovered in the code added to CARL III since the release of version 3, some in the CDC to VAX conversion code, and a few were discovered in the original version as well.

Module CAREIN was enhanced at NASA in the areas of the fault-tree calculations and the testing of user inputs. Several additional tests of user inputs were also added by Sequoia Systems, Inc.

As a result of this enhancement phase all separate versions of CARE III have been consolidated into one FORTRAN 77 standard version capable of being run on either a CDC or a VAX machine. This enhanced CARE III version 4 computer program is now a much more practical and useful engineering tool for predicting the unreliability of highly reliable reconfigurable fault-tolerant systems.

11.0 REFERENCES

1. Bavuso, S.J., J.E. Brunelle, and P.L. Petersen. "CARE III Hands-On Demonstration and Tutorial," NASA Technical Memorandum 85811, May 1984.

2. Bavuso, S.J. and P.L. Petersen. "CARE III Model Overview and User's Guide (first revision)," NASA Technical Memorandum 86404, April 1985.

3. Gear, C.W., "The Automatic Integration of Stiff Ordinary Differential Equations." In Information Processing 68, ed. A.J.H. Morrell. Amsterdam: North-Holland, pp. 187-193, 1969.

4. Hildebrand, Francis B., Methods of Applied Mathematics. Englewood Cliffs, N.J.: Prentice-Hall, 1965.

5. Rose, D.M., R.E. Altschul, J.W.Manke, and D.L. Nelson. "Review and Verification of CARE III Mathematical Model and Code: Interim Report," NASA Contractor Report 166096, April 1983.

6. Rose, D.M., J.W. Manke, R.E. Altschul, and D.L. Nelson. "Correction and Improvement of CARE III, Version 3," NASA Contractor Report 166122, April 1984.

7. Smotherman, Mark. "Parametric Error Analysis and Coverage Approximations in Reliability Modeling," Dissertation Defense, University of North Carolina at Chapel Hill, March 1984.

8. Stiffler, J.J. and L.A. Bryant. "CARE III Phase II Report - Mathematical Description," NASA Contractor Report 3566, November 1982.

9. Stiffler, J.J., J.S. Neumann, and L.A. Bryant. "CARE III Phase III Report - Test and Evaluation," NASA Contractor Report 3631, November 1982.

This Page Intentionally Left Blank

APPENDIX A

Symbolic Name Equates

CAREIN:

```
define(CARDOUT,CRDOUT);          define(CRTLPRS,CRTLPR);
define(LCNDVEC,LCNDVC);          define(MFLTYPE,MFLTYP);
define(MINTRMS,MNTRMS);          define(MSRVVEC,MSRVVC);
define(NCONVEC,NCONVC);          define(NFLTCAT,NFLTCT);
define(PROCIND,PRCIND);          define(RELSTEP,RELSTP)
```

COVRGE:

```
define(ALGTRNC,ALGTRC);          define(AREALST,ARELST);
define(ARMOMNT,ARMONT);          define(ARTOINT,ARTINT);
define(CNSINTG,CNSINT);          define(CNSRTDN,CNRTDN);
define(CNVLINT,CVLINT);          define(COMPFUN,CMPFUN);
define(CVGSTEP,CVGSTP);          define(CVRGCOM,CVRGCM);
define(DFM0LST,DFM0LS);          define(DFM1LST,DFM1LS);
define(DFM2LST,DFM2LS);          define(DFMSTEP,DFMSTP);
define(DFSTPSZ,DFSTSZ);          define(DIFCHNG,DIFCHG);
define(FCNVLTM,FCVLTM);          define(FEEINTF,FEINTF);
define(FEESTEP,FEESTP);          define(FEESTSZ,FESTSZ);
define(FEETMAR,FETMAR);          define(FILLDBL,FILDBL);
define(FILLSNG,FILSNG);          define(FLTSTEP,FLTSTP);
define(FLTYPCM,FLTPCM);          define(FRETVAL,FRETVL);
define(FSTPSZE,FSTPSZ);          define(FTCHSTP,FTCHST);
define(GENMNTS,GNMNTS);          define(GENTMAR,GNTMAR);
define(IAXSTYP,IAXSTP);          define(IDIVCNT,IDVCNT);
define(IFEEPNT,IFEPNT);          define(IHLDDUB,IHLDDB);
define(INDXPAR,IDXPAR);          define(ITLSTP1,ITLSP1);
define(ITREGMX,ITRGMX);          define(KNTSTPS,KNTSTP);
define(NDFMSTP,NDFMST);          define(NFEESTP,NFEEST);
define(NFLTSTP,NFLTST);          define(NFSTPAR,NFSTAR);
define(NGSTPAR,NGSTAR);          define(NPARSTP,NPARST);
define(NPB1STP,NPB1ST);          define(NPB2STP,NPB2ST);
define(NPBGSTP,NPBGST);          define(NPDFSTP,NPDFST);
define(NPDPSTP,NPDPST);          define(NPERSTP,NPERST);
define(NPLTSTP,NPLTST);          define(NPNBSTP,NPNBST);
define(NPNESTP,NPNEST);          define(NPSASTP,NPSAST);
define(NPSBSTP,NPSBST);          define(NPSTPIN,NPSTIN);
define(NSFMSTP,NSFMST);          define(NSTEPAR,NSTPAR);
define(NSTPAR1,NSTAR1);          define(NSTPAR2,NSTAR2);
define(NVNRSTP,NVNRST);          define(NXB1STP,NXB1ST);
define(NXB2STP,NXB2ST);          define(NZROSTP,NZROST);
define(PARSTEP,PARSTP);          define(PARTMAR,PARTAR);
define(PB1STEP,PB1STP);          define(PB2STEP,PB2STP);
define(PBGSTEP,PBGSTP);          define(PDFSTEP,PDFSTP);
define(PDPSTEP,PDPSTP);          define(PEAKFLG,PEAKFL);
define(PERSTEP,PERSTP);          define(PFSMTMX,PFSMTX);
```

(COVRGE name equates continued)

```
define(PINTFLG,PINTFL);          define(PLTSTEP,PLTSTP);
define(PNBSTEP,PNBSTP);          define(PNESTEP,PNESTP);
define(PREVNRC,PRVNRC);          define(PRNTCVG,PRNTCV);
define(PSASTEP,PSASTP);          define(PSBSTEP,PSBSTP);
define(PSTPINC,PSTINC);          define(PSTPRNT,PSTPRT);
define(PSTPSZE,PSTPSZ);          define(PXSMDEN,PXSMDN);
define(PXSMNUM,PXSMNM);          define(RATECOM,RATECM);
define(REALMAX,REALMX);          define(REALMIN,REALMN);
define(RECRSVF,RCRSVF);          define(REGINTG,REGINT);
define(RELSTEP,RELSTP);          define(RELTIME,RELTME);
define(RTDNINT,RTDNIN);          define(SFM0LST,SFM0LS);
define(SFM1LST,SFM1LS);          define(SFM2LST,SFM2LS);
define(SFMSTEP,SFMSTP);          define(SFSTPST,SFSTST);
define(SFSTPSZ,SFSTSZ);          define(SIMPINT,SMPINT);
define(STDYDIF,STDYDF);          define(STDYFLG,STDYFL);
define(STEP1ST,STP1ST);          define(STEP2ST,STP2ST);
define(STPINDX,STINDX);          define(STPINIT,STINIT);
define(STPLSTH,STLSTH);          define(STPSTRT,STSTRT);
define(SUMPFEE,SMPFEE);          define(TADRMXI,TADRMI);
define(TADRMXJ,TADRMJ);          define(TMARCOM,TMARCM);
define(TMAXDBL,TMXDBL);          define(TMAXSNG,TMXSNG);
define(TMPNTRS,TMPNTR);          define(TWOPWR7,TWOPR7);
define(TZEROST,TZROST);          define(VLTNREC,VLTNRC);
define(VNRSTEP,VNRSTP);          define(VOLTERA,VOLTRA);
define(VSTPINT,VSTINT);          define(XB1INTG,XB1INT);
define(XB1STEP,XB1STP);          define(XB2INTG,XB2INT);
define(XB2STEP,XB2STP);          define(XFNATIM,XFATTM);
define(ZROSTEP,ZROSTP)
```

CVGPLT:

```
define(AEPZFLG,AEPZFG);          define(ARTOPLT,ARTPLT);
define(AXESLOG,AXSLOG);          define(AZROFLG,AZROFG);
define(CAREPLT,CREPLT);          define(CVGSTEP,CVGSTP);
define(DBLFUNC,DBLFNC);          define(DIFCHNG,DIFCHG);
define(GENTMAR,GNTMAR);          define(GENXPTS,GNXPTS);
define(IAXSTYP,IAXSTP);          define(NCYCLEY,NCYCLY);
define(NPBGSTP,NPBGST);          define(NPDFSTP,NPDFST);
define(NPDPSTP,NPDPST);          define(NPLTSTP,NPLTST);
define(NPNBSTP,NPNBST);          define(NSTEPAR,NSTPAR);
define(PBGSTEP,PBGSTP);          define(PDFSTEP,PDFSTP);
define(PDPSTEP,PDPSTP);          define(PLTSTEP,PLTSTP);
define(PNBSTEP,PNBSTP);          define(PSTPSZE,PSTPSZ);
define(RELSTEP,RELSTP);          define(SNGFUNC,SNGFNC);
define(STPINDX,STINDX);          define(XAXISMX,XAXSMX);
define(XZROFLG,XZROFG);          define(YAXISMX,YAXSMX)
```

CARE3:

```
define(ARTOINT,ARTINT);          define(BUFFOUT,BUFOUT);
define(CNFGVAR,CNFGVR);          define(COMPPST,CMPPST);
define(CVRGCOM,CVRGCM);          define(FINTGRT,FINTGT);
define(FMTSTGS,FMTSTG);          define(FPSTREC,FPSTRC);
define(GHSFPTS,GHSFPT);          define(GNFLTVC,GNFLVC);
define(GNVCCOM,GNVCCM);          define(HBNGVAR,HBNGVR);
define(IFLTVEC,IFLTVC);          define(IJSTGIN,IJSGIN);
define(ISETVCS,ISTVCS);          define(ISTINIT,ISTNIT);
define(ISTSTOP,ISTSTP);          define(ITBLAST,ITBLST);
define(ITBLKSZ,ITBKSZ);          define(KMlINDX,KMlIDX);
define(KPlINDX,KPlIDX);          define(LCNDVEC,LCNDVC);
define(LFLTVEC,LFLTVC);          define(LMUNTVC,LMUTVC);
define(LVECTOR,LVCTOR);          define(MAXLAST,MXLAST);
define(MFLTYPE,MFLTYP);          define(MLTPLYR,MLTPLY);
define(MSRVVEC,MSRVVC);          define(NCONVEC,NCONVC);
define(NFLTCAT,NFLTCT);          define(NFLTVDP,NFLTDP);
define(NONLDEP,NONLDP);          define(PNOEFCT,PNOEFC);
define(PRNTPST,PRTPST);          define(QINTGRL,QINTGL);
define(QNOEFCT,QNOEFC);          define(REALMAX,REALMX);
define(RELSTEP,RELSTP);          define(STEPCOM,STEPCM);
define(STOPARM,STOPRM);          define(TRNSFLT,TRNFLT);
define(TZEROST,TZROST)
```

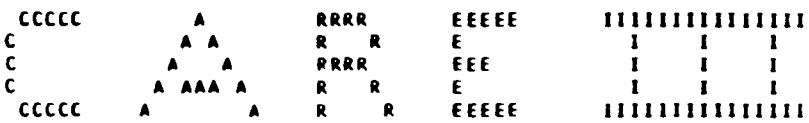
RELPLT:

```
define(AEPZFLG,AEPZFG);          define(ARTOPLT,ARTPLT);
define(AXLSLOG,AXSLOG);          define(AZROFLG,AZROFG);
define(CAREPLT,CREPLT);          define(GENXPTS,GNXPTS);
define(IAXSTYP,IAXSTP);          define(NCYCLEY,NCYCLY);
define(RELSTEP,RELSTP);          define(STEPCOM,STEPCM);
define(XAXISMX,XAXSMX);          define(XZROFLG,XZROFG);
define(YAXISMX,YAXSMX)
```

This Page Intentionally Left Blank

APPENDIX B

Example Problem 5
Output Listings



*** CARE III Workshop (with fault types switched) Example Problem 5 ***

SUBRUN 1

STAGES 1 - 6

CONFIGURATION:

STAGE	N	M
1	3	2
2	3	2
3	4	2
4	1	1
5	1	1
6	3	2

FAULT INFORMATION:

ICAT	RLM	OMG	JTYP
1	1.500e-05	1.00e+00	4
1	1.900e-05	1.00e+00	4
1	7.200e-03	1.00e+00	3
2	3.300e-04	1.00e+00	1
3	4.800e-04	1.00e+00	2
1	1.700e-10	1.00e+00	4
1	2.300e-09	1.00e+00	4
1	3.700e-05	1.00e+00	4

FAULT VECTORS:

TIMES AT WHICH THE FOLLOWING FUNCTION VALUES CORRESPOND (IN HRS):

0.0000000000e+00 2.0000000298e-01 4.0000000596e-01 6.0000002384e-01 8.0000001192e-01
1.0000000000e+00 1.2000000477e+00 1.4000000954e+00 1.6000001431e+00 1.8000001907e+00
2.0000002384e+00 2.2000002861e+00 2.4000003338e+00 2.6000003815e+00 2.8000004292e+00
3.0000004768e+00 3.2000005245e+00 3.4000005722e+00 3.6000006199e+00 3.8000006676e+00
4.0000004768e+00 4.2000002861e+00 4.4000000954e+00 4.5999999046e+00 4.7999997139e+00
4.9999995232e+00 5.1999993324e+00 5.3999991417e+00 5.5999989510e+00 5.7999987602e+00
5.9999985695e+00 6.1999983788e+00 6.3999981880e+00 6.5999979973e+00 6.7999978065e+00
6.9999976158e+00 7.1999974251e+00 7.3999972343e+00 7.5999970436e+00 7.7999968529e+00
7.9999966621e+00 8.1999969482e+00 8.3999967575e+00 8.5999965668e+00 8.7999963760e+00
8.9999961853e+00 9.1999959946e+00 9.3999958038e+00 9.5999956131e+00 9.7999954224e+00
9.9999952316e+00

```
                .
                .
                .
SUMMARY  INFORMATION:

Q  SUM  =
          0.0000000000e+00 4.7354771482e-07 1.2194415149e-06 1.9402057205e-06 2.6661437005e-06
          3.3916994653e-06 4.1168768803e-06 4.8416727623e-06 5.5660848375e-06 6.2901176534e-06
          7.0137707553e-06 7.7370459621e-06 8.4599414549e-06 9.1824540505e-06 9.9045892057e-06
          1.0626343283e-05 1.1347720829e-05 1.2068719116e-05 1.2789337234e-05 1.3509577911e-05
          1.4229441149e-05 1.4948925127e-05 1.5668030755e-05 1.6386758944e-05 1.7105114239e-05
          1.7823087546e-05 1.8540684323e-05 1.9257911845e-05 1.9974753741e-05 2.0691222744e-05
          2.1407317035e-05 2.2123031158e-05 2.2838374207e-05 2.3553340725e-05 2.4267936169e-05
          2.4982149625e-05 2.5695993827e-05 2.6409463317e-05 2.7122556276e-05 2.7835278161e-05
          2.8547619877e-05 2.9259595976e-05 2.9971195545e-05 3.0682418583e-05 3.1393276004e-05
          3.2103762351e-05 3.2813870348e-05 3.3523610909e-05 3.4232976759e-05 3.4941967897e-05
          3.5650587961e-05

P* SUM  =
          0.0000000000e+00 2.3459759135e-10 9.3838059545e-10 2.1113346627e-09 3.7534442221e-09
          5.8646958401e-09 8.4450748616e-09 1.1494567076e-08 1.5013155164e-08 1.9000831131e-08
          2.3457573661e-08 2.8383370321e-08 3.3778199793e-08 3.9642060301e-08 4.5974928753e-08
          5.2776794490e-08 6.0047646855e-08 6.7787453872e-08 7.5996211990e-08 8.4673935419e-08
          9.3820531788e-08 1.0343606505e-07 1.1352047835e-07 1.2407377881e-07 1.3509591668e-07
          1.4658692749e-07 1.5854675439e-07 1.7097541161e-07 1.8387287071e-07 1.9723910327e-07
          2.1107418036e-07 2.2537794564e-07 2.4015045597e-07 2.5539173976e-07 2.7110164069e-07
          2.8728044299e-07 3.0392772032e-07 3.2104378533e-07 3.3862858118e-07 3.5668188048e-07
          3.7520385376e-07 3.9419447262e-07 4.1365368020e-07 4.3358153334e-07 4.5397780468e-07
          4.7484277843e-07 4.9617608511e-07 5.1797815104e-07 5.4024866358e-07 5.6298762274e-07
          5.8619508536e-07

Q*P* SUM  =
          0.0000000000e+00 4.7378230761e-07 1.2203798860e-06 1.9423171125e-06 2.6698971851e-06
          3.3975641145e-06 4.1253219933e-06 4.8531674111e-06 5.5810978665e-06 6.3091183620e-06
          7.0372284426e-06 7.7654294728e-06 8.4937200882e-06 9.2220961960e-06 9.9505641629e-06
          1.0679120351e-05 1.1407768397e-05 1.2136506484e-05 1.2865333701e-05 1.3594251868e-05
          1.4323261894e-05 1.5052361050e-05 1.5781552065e-05 1.6510832211e-05 1.7240210582e-05
          1.7969674445e-05 1.8699231077e-05 1.9428887754e-05 2.0158626285e-05 2.0888461222e-05
          2.1618390747e-05 2.2348409402e-05 2.3078524464e-05 2.3808732294e-05 2.4539038350e-05
          2.5269429898e-05 2.5999921490e-05 2.6730507670e-05 2.7461184800e-05 2.8191960155e-05
          2.8922824640e-05 2.9653790989e-05 3.0384850106e-05 3.1116000173e-05 3.1847252103e-05
          3.2578605897e-05 3.3310047002e-05 3.4041589970e-05 3.4773223888e-05 3.5504956031e-05
          3.6236782762e-05



NUMBER OF FAILED        UNRELIABILITY AT        PERFECT COVERAGE UNRELIABILITY
     STAGES               10.0000  HRS          AT   10.0000  HRS

       0                 3.5650275095e-05        0.0000000000e+00
       1                 3.1313029947e-10        5.8619343690e-07
       2                 X                       1.6546414846e-12

TOTAL SYSTEM UNRELIABILITY AT   10.0000  HRS = 3.6236782762e-05
```

Example Problem 5: FT = 10 hr. - Equal Steps
RELSTP = 0.2 hr.
NSTEPS = 50

```
 CCCCC         A         RRRR      EEEEE      IIIIIIIIIIIIIII
C             A A        R   R     E            I    I    I
C            A   A       RRRR      EEE          I    I    I
C           A AAA A      R   R     E            I    I    I
 CCCCC     A       A     R    R    EEEEE      IIIIIIIIIIIIIII
```

*** CARE III Workshop (with fault types switched) Example Problem 5 ***

SUBRUN 1

STAGES 1 - 6

CONFIGURATION:

STAGE	N	M
1	3	2
2	3	2
3	4	2
4	1	1
5	1	1
6	3	2

FAULT INFORMATION:

ICAT	RLR	OMG	JTYP
1	1.500e-05	1.00e+00	4
1	1.900e-05	1.00e+00	4
1	7.200e-03	1.00e+00	3
2	3.300e-04	1.00e+00	1
3	4.800e-04	1.00e+00	2
1	1.700e-10	1.00e+00	4
1	2.300e-09	1.00e+00	4
1	3.700e-05	1.00e+00	4

FAULT VECTORS:

TIMES AT WHICH THE FOLLOWING FUNCTION VALUES CORRESPOND (IN HRS):
0.0000000000e+00 9.7703956999e-04 1.9540791400e-03 2.9311187100e-03 3.9081582800e-03
4.8851980828e-03 6.8392772228e-03 8.7933568284e-03 1.0747436434e-02 1.2701516040e-02
1.4655595645e-02 1.8563754857e-02 2.2471914068e-02 2.6380073279e-02 3.0288232490e-02
3.4196391702e-02 4.2012710124e-02 4.9829028547e-02 5.7645346969e-02 6.5461665392e-02
7.3277980089e-02 8.8910616934e-02 1.0454325378e-01 1.2017589062e-01 1.3580852747e-01
1.5144115686e-01 1.8270643055e-01 2.1397170424e-01 2.4523697793e-01 2.7650225163e-01
3.0776751041e-01 3.7029805779e-01 4.3282860518e-01 4.9535915256e-01 5.5788969994e-01
6.2042021751e-01 7.4548131227e-01 8.7054240704e-01 9.9560350180e-01 1.1206645966e+00
1.2457256317e+00 1.4958478212e+00 1.7459700108e+00 1.9960922003e+00 2.2462143898e+00
2.4963364601e+00 2.9965808392e+00 3.4968252182e+00 3.9970695972e+00 4.4973139763e+00
4.9975581169e+00 5.9980468750e+00 6.9985356331e+00 7.9990243912e+00 8.9995126724e+00
1.0000000954e+01

.
.

```
S U M M A R Y   I N F O R M A T I O N :

Q  SUM  =
          0.0000000000e+00 4.6099676854e-10 1.8523756973e-09 3.9527217055e-09 6.5079350797e-09
          9.3664729306e-09 1.5626751448e-08 2.2289745161e-08 2.9155335923e-08 3.6126500902e-08
          4.3155964136e-08 5.7298400691e-08 7.1486297770e-08 8.5689222828e-08 9.9897164318e-08
          1.1410666900e-07 1.4252655944e-07 1.7094622251e-07 1.9936531714e-07 2.2778382913e-07
          2.5620175848e-07 3.1303585502e-07 3.6986762098e-07 4.2669697109e-07 4.8352404747e-07
          5.4034870800e-07 6.5399092364e-07 7.6762387380e-07 8.8124738795e-07 9.9486169347e-07
          1.1084665630e-06 1.3356485624e-06 1.5627928178e-06 1.7898997839e-06 2.0169693471e-06
          2.2440019620e-06 2.6979548693e-06 3.1517586194e-06 3.6054134398e-06 4.0589188757e-06
          4.5122756092e-06 5.4185411500e-06 6.3242109718e-06 7.2292868936e-06 8.1337693700e-06
          9.0376588560e-06 1.0843656128e-05 1.2647282347e-05 1.4448543880e-05 1.6247435269e-05
          1.8043971068e-05 2.1629981347e-05 2.5206576538e-05 2.8773805752e-05 3.2331688999e-05
          3.5880242649e-05

P* SUM  =
          0.0000000000e+00 5.5987658086e-15 2.2395063234e-14 5.0388885501e-14 8.9580239385e-14
          1.3996913166e-13 2.7433944331e-13 4.5349994918e-13 6.7745050021e-13 9.4619115060e-13
          1.2597217106e-12 2.0211534269e-12 2.9617449986e-12 4.0814968592e-12 5.3804079247e-12
          6.8584794961e-12 1.0352100253e-11 1.4562358264e-11 1.9489253961e-11 2.5132788212e-11
          3.1492950608e-11 4.6363187595e-11 6.4099954511e-11 8.4703244418e-11 1.0817304691e-10
          1.3450934810e-10 1.9578147437e-10 2.6851956769e-10 3.5272354482e-10 4.4839332247e-10
          5.5552890066e-10 8.0419715331e-10 1.0987281085e-09 1.4391208225e-09 1.8253751843e-09
          2.2574906389e-09 3.2593032717e-09 4.4445549463e-09 5.8132414438e-09 7.3653620980e-09
          9.1009084713e-09 1.3122275710e-08 1.7877312075e-08 2.3365991808e-08 2.9588283823e-08
          3.6544161475e-08 5.2656556448e-08 7.1702949356e-08 9.3683105717e-08 1.1859682303e-07
          1.4644382418e-07 2.1093688929e-07 2.8716044653e-07 3.7511267692e-07 4.7479170462e-07
          5.8619576748e-07

Q+P* SUM  =
          0.0000000000e+00 4.6100237516e-10 1.8523981238e-09 3.9527718876e-09 6.5080247857e-09
          9.3666132628e-09 1.5627025007e-08 2.2290198132e-08 2.9156012715e-08 3.6127445924e-08
          4.3157225349e-08 5.7300422185e-08 7.1489260733e-08 8.5693301344e-08 9.9902543127e-08
          1.1411352574e-07 1.4253690495e-07 1.7096078864e-07 1.9938480023e-07 2.2780896813e-07
          2.5623324973e-07 3.1308221082e-07 3.6993171193e-07 4.2678166778e-07 4.8363222049e-07
          5.4048319953e-07 6.5418669237e-07 7.6789240211e-07 8.8160010137e-07 9.9531007436e-07
          1.1090220369e-06 1.3364527831e-06 1.5638916011e-06 1.7913389456e-06 2.0187947030e-06
          2.2462595552e-06 2.7012142709e-06 3.1562030927e-06 3.6112267026e-06 4.0662844185e-06
          4.5213764679e-06 5.4316633396e-06 6.3420884544e-06 7.2526527219e-06 8.1633579612e-06
          9.0742032626e-06 1.0896312233e-05 1.2718985090e-05 1.4542227291e-05 1.6366031559e-05
          1.8190414266e-05 2.1840918635e-05 2.5493736757e-05 2.9148917747e-05 3.2806481613e-05
          3.6466437450e-05


NUMBER OF FAILED STAGES    UNRELIABILITY AT 10.0000 HRS    PERFECT COVERAGE UNRELIABILITY AT 10.0000 HRS

          0                3.5879929783e-05                0.0000000000e+00
          1                3.1332850203e-10                5.8619411902e-07
          2                X                               1.6546433277e-12

TOTAL SYSTEM UNRELIABILITY AT  10.0000  HRS = 3.6466437450e-05
```

Example Problem 5: FT = 10 hr. - Unequal Steps
CVGSTP = 9.7704e - 4 hr.
NPERST = 5
NDUB = 10
NSTEPS = 55

```
 CCCCC         A        RRRR      EEEEE      IIIIIIIIIIIIIII
C             A A       R   R     E            I    I    I
C            A   A      RRRR      EEE          I    I    I
C           A AAA A     R   R     E            I    I    I
 CCCCC     A       A    R    R    EEEEE      IIIIIIIIIIIIIII
```

*** CARE III Workshop (with fault types switched) Example Problem 5 ***

S U B R U N 1

S T A G E S 1 - 6

C O N F I G U R A T I O N :

S T A G E	N	M
1	3	2
2	3	2
3	4	2
4	1	1
5	1	1
6	3	2

F A U L T I N F O R M A T I O N :

I C A T	R L M	O M G	J T Y P
1	1.500e-05	1.00e+00	4
1	1.900e-05	1.00e+00	4
1	7.200e-03	1.00e+00	3
2	3.300e-04	1.00e+00	1
3	4.800e-04	1.00e+00	2
1	1.700e-10	1.00e+00	4
1	2.300e-09	1.00e+00	4
1	3.700e-05	1.00e+00	4

F A U L T V E C T O R S :

TIMES AT WHICH THE FOLLOWING FUNCTION VALUES CORRESPOND (IN HRS):

0.0000000000e+00 3.2000000000e+01 6.4000000000e+01 9.6000000000e+01 1.2800000000e+02
1.6000000000e+02 1.9200000000e+02 2.2400000000e+02 2.5600000000e+02 2.8800000000e+02
3.2000000000e+02 3.5200000000e+02 3.8400000000e+02 4.1600000000e+02 4.4800000000e+02
4.8000000000e+02 5.1200000000e+02 5.4400000000e+02 5.7600000000e+02 6.0800000000e+02
6.4000000000e+02 6.7200000000e+02 7.0400000000e+02 7.3600000000e+02 7.6800000000e+02
8.0000000000e+02 8.3200000000e+02 8.6400000000e+02 8.9600000000e+02 9.2800000000e+02
9.6000000000e+02 9.9200000000e+02 1.0240000000e+03 1.0560000000e+03 1.0880000000e+03
1.1200000000e+03 1.1520000000e+03 1.1840000000e+03 1.2160000000e+03 1.2480000000e+03
1.2800000000e+03 1.3120000000e+03 1.3440000000e+03 1.3760000000e+03 1.4080000000e+03
1.4400000000e+03 1.4720000000e+03 1.5040000000e+03 1.5360000000e+03 1.5680000000e+03
1.6000000000e+03

.
.

SUMMARY INFORMATION:

Q SUM =

0.0000000000e+00 7.1052709245e-05 1.7681167810e-04 2.7046215837e-04 3.5756084253e-04
4.3786133756e-04 5.1194627304e-04 5.8034574613e-04 6.4354011556e-04 7.0196646266e-04
7.5602228753e-04 8.0606975826e-04 8.5243786452e-04 8.9542707428e-04 9.3531073071e-04
9.7233842826e-04 1.0067375842e-03 1.0387166403e-03 1.0684658773e-03 1.0961579392e-03
1.1219526641e-03 1.1459956877e-03 1.1684194906e-03 1.1893478222e-03 1.2088909280e-03
1.2271528831e-03 1.2442275183e-03 1.2602016795e-03 1.2751553440e-03 1.2891620863e-03
1.3022894273e-03 1.3145996490e-03 1.3261499116e-03 1.3369928347e-03 1.3471777784e-03
1.3567500282e-03 1.3657507952e-03 1.3742189622e-03 1.3821898028e-03 1.3896971941e-03
1.3967704726e-03 1.4034383930e-03 1.4097280800e-03 1.4156621182e-03 1.4212656533e-03
1.4265576610e-03 1.4315581648e-03 1.4362862566e-03 1.4407574199e-03 1.4449890004e-03
1.4489939203e-03

P* SUM =

0.0000000000e+00 5.9955941651e-06 2.3943121050e-05 5.3783431213e-05 9.5455630799e-05
1.4890173043e-04 2.1406181622e-04 2.9087648727e-04 3.7928784150e-04 4.7923365491e-04
5.9065938694e-04 7.1350333747e-04 8.4770238027e-04 9.9320767913e-04 1.1499531101e-03
1.3178786030e-03 1.4969247859e-03 1.6870428808e-03 1.8881669967e-03 2.1002346184e-03
2.3231951054e-03 2.5569722056e-03 2.8015277348e-03 3.0567988288e-03 3.3227321692e-03
3.5992409103e-03 3.8862982765e-03 4.1838334873e-03 4.4917906635e-03 4.8101074062e-03
5.1387259737e-03 5.4776030593e-03 5.8266604319e-03 6.1858436093e-03 6.5551032312e-03
6.9343824871e-03 7.3236264288e-03 7.7227642760e-03 8.1317294389e-03 8.5505163297e-03
8.9790001512e-03 9.4171706587e-03 9.8649533466e-03 1.0322293267e-02 1.0789157823e-02
1.1265436187e-02 1.1751125567e-02 1.2246135622e-02 1.2750442140e-02 1.3263940811e-02
1.3786630705e-02

Q*P* SUM =

0.0000000000e+00 7.7048302046e-05 2.0075480279e-04 3.2424560050e-04 4.5301648788e-04
5.8676308254e-04 7.2600808926e-04 8.7122223340e-04 1.0228279280e-03 1.1812001467e-03
1.3466817327e-03 1.5195731539e-03 1.7001403030e-03 1.8886347534e-03 2.0852638409e-03
2.2902169731e-03 2.5036623701e-03 2.7257595211e-03 2.9566329904e-03 3.1963926740e-03
3.4451477695e-03 3.7029678933e-03 3.9699473418e-03 4.2461468838e-03 4.5316233300e-03
4.8263939098e-03 5.1305256784e-03 5.4440353997e-03 5.7669458911e-03 6.0992697254e-03
6.4410152845e-03 6.7922025919e-03 7.1528102271e-03 7.5228363276e-03 7.9022813588e-03
8.2911327481e-03 8.6893774569e-03 9.0969828889e-03 9.5139192417e-03 9.9402135238e-03
1.0375770740e-02 1.0820609517e-02 1.1274681427e-02 1.1737955734e-02 1.2210423127e-02
1.2691994198e-02 1.3182683848e-02 1.3682422228e-02 1.4191200025e-02 1.4708929695e-02
1.5235624276e-02

NUMBER OF FAILED STAGES	UNRELIABILITY AT 1600.0000 HRS	PERFECT COVERAGE UNRELIABILITY AT 1600.0000 HRS
0	1.3569903094e-03	0.0000000000e+00
1	9.2003952886e-05	3.5010457505e-03
2	1.5216904546e-16	1.0251815431e-02
3	X	3.3737782360e-05
4	X	3.1090326758e-08

TOTAL SYSTEM UNRELIABILITY AT 1600.0000 HRS = 1.5235624276e-02

Example Problem 5: FT = 1600 hr. - Equal Steps
RELSTP = 32.0 hr.
NSTEPS = 50

```
 CCCCC         A         RRRR      EEEEE      IIIIIIIIIIIIIII
C             A A        R   R     E              I     I     I
C            A   A       RRRR      EE             I     I     I
C           A AAA A      R   R     E              I     I     I
 CCCCC     A       A     R    R    EEEEE      IIIIIIIIIIIIIII
```

*** CARE III Workshop (with fault types switched) Example Problem 5 ***

SUBRUN 1

STAGES 1 - 6

CONFIGURATION:

STAGE	N	M
1	3	2
2	3	2
3	4	2
4	1	1
5	1	1
6	3	2

FAULT INFORMATION:

ICAT	RLM	OMG	JTYP
1	1.500e-05	1.00e+00	4
1	1.900e-05	1.00e+00	4
1	7.200e-03	1.00e+00	3
2	3.300e-04	1.00e+00	1
3	4.800e-04	1.00e+00	2
1	1.700e-10	1.00e+00	4
1	2.300e-09	1.00e+00	4
1	3.700e-05	1.00e+00	4

FAULT VECTORS:

TIMES AT WHICH THE FOLLOWING FUNCTION VALUES CORRESPOND (IN HRS):

```
0.0000000000e+00 1.0172545444e-03 2.0345090888e-03 3.0517636333e-03 5.0862729549e-03
7.1207820438e-03 9.1552911326e-03 1.3224309310e-02 1.7293328419e-02 2.1362347528e-02
2.9500383884e-02 3.7638422102e-02 4.5776460320e-02 6.2052533031e-02 7.8328609467e-02
9.4604685903e-02 1.2715683877e-01 1.5970899165e-01 1.9226114452e-01 2.5736543536e-01
3.2246974111e-01 3.8757404685e-01 5.1778262854e-01 6.4799124002e-01 7.7819985151e-01
1.0386170149e+00 1.2990342379e+00 1.5594514608e+00 2.0802857876e+00 2.6011202335e+00
3.1219546795e+00 4.1636233330e+00 5.2052922249e+00 6.2469611168e+00 8.3302984238e+00
1.0413636208e+01 1.2496973991e+01 1.6663648605e+01 2.0830324173e+01 2.4996999741e+01
3.3330348969e+01 4.1663700104e+01 4.9997051239e+01 6.6663749695e+01 8.3330451965e+01
9.9997154236e+01 1.3333055115e+02 1.6666395569e+02 1.9999736023e+02 2.6666415405e+02
3.3333096313e+02 3.9999777222e+02 5.3333135986e+02 6.6666497803e+02 7.9999859619e+02
1.0666657715e+03 1.3333330078e+03 1.6000002441e+03
```

.
.

SUMMARY INFORMATION:

Q SUM =

0.0000000000e+00 5.1062903905e-10 2.0075212603e-09 4.2477088513e-09 9.9855483882e-09
1.6583147300e-08 2.3569311836e-08 3.8020264270e-08 5.2708049481e-08 6.7470068643e-08
9.7045358416e-08 1.2663377902e-07 1.5622330807e-07 2.1540077455e-07 2.7457579677e-07
3.3374828945e-07 4.5208568622e-07 5.7041296486e-07 6.8872998327e-07 9.2533360885e-07
1.1618968756e-06 1.3984192719e-06 1.8713429881e-06 2.3441048143e-06 2.8167048640e-06
3.7614208850e-06 4.7054886636e-06 5.6489097915e-06 7.5338252827e-06 9.4161650850e-06
1.1295929653e-05 1.5047755369e-05 1.8789331079e-05 2.2520696803e-05 2.9952903787e-05
3.7344638258e-05 4.4696109399e-05 5.9279278503e-05 7.3704344686e-05 8.7973188784e-05
1.1604958127e-04 1.4352309518e-04 1.7040793318e-04 2.2246678418e-04 2.7233306901e-04
3.2010822906e-04 4.0976624587e-04 4.9215706531e-04 5.6792778196e-04 7.0189376129e-04
8.1571267219e-04 9.1270747362e-04 1.0666849557e-03 1.1804752285e-03 1.2655236060e-03
1.3788309880e-03 1.4461012324e-03 1.4872917673e-03

P* SUM =

0.0000000000e+00 6.0691418389e-15 2.4276567356e-14 5.4622271468e-14 1.5172853750e-13
2.9738784677e-13 4.9160030430e-13 1.0256843118e-12 1.7539805329e-12 2.6764892386e-12
5.1041419355e-12 8.3086411015e-12 1.2289986737e-11 2.2583209608e-11 3.5983809682e-11
5.2491771346e-11 9.4829762098e-11 1.4959711248e-10 2.1679376350e-10 3.8847461359e-10
6.0987204176e-10 8.8098534023e-10 1.5723575775e-09 2.4625876893e-09 3.5516714014e-09
6.3263834171e-09 9.8964623163e-09 1.4261874348e-08 2.5378572133e-08 3.9676216090e-08
5.7154551314e-08 1.0165226172e-07 1.5886962501e-07 2.2880462325e-07 4.0681914015e-07
6.3567927100e-07 9.1536878699e-07 1.6271687855e-06 2.5420872589e-06 3.6599940358e-06
6.5041081143e-06 1.0158786608e-05 1.4622553863e-05 2.5974455639e-05 4.0550366975e-05
5.8342848206e-05 1.0354306141e-04 1.6150622105e-04 2.3217085982e-04 4.1131704347e-04
6.4045010367e-04 9.1903610155e-04 1.6224342398e-03 2.5172531605e-03 3.5992308985e-03
6.3078058884e-03 9.7146183252e-03 1.3786630705e-02

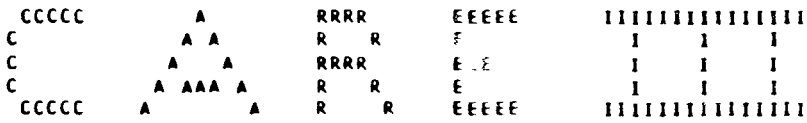
Q*P* SUM =

0.0000000000e+00 5.1063508977e-10 2.0075454632e-09 4.2477634743e-09 9.9857002667e-09
1.6583443951e-08 2.3569803886e-08 3.8021291004e-08 5.2709804521e-08 6.7472747389e-08
9.7050460113e-08 1.2664209237e-07 1.5623560046e-07 2.1542335560e-07 2.7461177865e-07
3.3380078435e-07 4.5218052946e-07 5.7056257674e-07 6.8894678407e-07 9.2572207677e-07
1.1625066918e-06 1.3993002312e-06 1.8729153908e-06 2.3465674985e-06 2.8202564408e-06
3.7677473301e-06 4.7153853302e-06 5.6631715779e-06 7.5592038229e-06 9.4558408819e-06
1.1353084119e-05 1.5149407773e-05 1.8948201614e-05 2.2749502023e-05 3.0359722587e-05
3.7980316847e-05 4.5611479436e-05 6.0906448198e-05 7.6246433309e-05 9.1633184638e-05
1.2255368347e-04 1.5368188906e-04 1.8503048341e-04 2.4844123982e-04 3.1288343598e-04
3.7845107727e-04 5.1330932183e-04 6.5366330091e-04 8.0009864178e-04 1.1132108048e-03
1.4561627759e-03 1.8317436334e-03 2.6891191956e-03 3.6977285054e-03 4.8647546209e-03
7.6866368763e-03 1.1160719208e-02 1.5273922123e-02

NUMBER OF FAILED STAGES	UNRELIABILITY AT 1600.0000 HRS	PERFECT COVERAGE UNRELIABILITY AT 1600.0000 HRS
0	1.3952857116e-03	0.0000000000e+00
1	9.2006332125e-05	3.5010441206e-03
2	1.5217402178e-16	1.0251816362e-02
3	X	3.3737793274e-05
4	X	3.1090330310e-08

TOTAL SYSTEM UNRELIABILITY AT 1600.0000 HRS = 1.5273922123e-02

Example Problem 5: FT = 1600 hr. - Unequal Steps
CVGSTP = 1.0173e-3 hr.
NPERST = 3
NDUB = 18
NSTEPS = 57



*** CARE III Workshop (with fault types switched) Example Problem 5 ***

SUBRUN 1

STAGES 1 - 6

CONFIGURATION:

STAGE	N	M
1	3	2
2	3	2
3	4	2
4	1	1
5	1	1
6	3	2

FAULT INFORMATION:

ICAT	RLM	OMG	JTYP
1	1.500e-05	1.00e+00	4
1	1.900e-05	1.00e+00	4
1	7.200e-03	1.00e+00	3
2	3.300e-04	1.00e+00	1
3	4.800e-04	1.00e+00	2
1	1.700e-10	1.00e+00	4
1	2.300e-09	1.00e+00	4
1	3.700e-05	1.00e+00	4

FAULT VECTORS:

TIMES AT WHICH THE FOLLOWING FUNCTION VALUES CORRESPOND (IN HRS):

0.0000000000e+00 1.6000000000e+03 3.2000000000e+03 4.8000000000e+03 6.4000000000e+03
8.0000000000e+03 9.6000000000e+03 1.1200000000e+04 1.2800000000e+04 1.4400000000e+04
1.6000000000e+04 1.7600000000e+04 1.9200000000e+04 2.0800000000e+04 2.2400000000e+04
2.4000000000e+04 2.5600000000e+04 2.7200000000e+04 2.8800000000e+04 3.0400000000e+04
3.2000000000e+04 3.3600000000e+04 3.5200000000e+04 3.6800000000e+04 3.8400000000e+04
4.0000000000e+04 4.1600000000e+04 4.3200000000e+04 4.4800000000e+04 4.6400000000e+04
4.8000000000e+04 4.9600000000e+04 5.1200000000e+04 5.2800000000e+04 5.4400000000e+04
5.6000000000e+04 5.7600000000e+04 5.9200000000e+04 6.0800000000e+04 6.2400000000e+04
6.4000000000e+04 6.5600000000e+04 6.7200000000e+04 6.8800000000e+04 7.0400000000e+04
7.2000000000e+04 7.3600000000e+04 7.5200000000e+04 7.6800000000e+04 7.8400000000e+04
8.0000000000e+04

```
.
.
.
S U M M A R Y   I N F O R M A T I O N :

Q  SUM    =
          0.0000000000e+00 1.6809122462e-04 2.7384635177e-04 2.7405927540e-04 2.7655158192e-04
          2.7696459438e-04 2.7705088723e-04 2.7706974652e-04 2.7707382105e-04 2.7707475238e-04
          2.7707492700e-04 2.7707498521e-04 2.7707498521e-04 2.7707498521e-04 2.7707498521e-04
          2.7707498521e-04 2.7707498521e-04 2.7707498521e-04 2.7707498521e-04 2.7707498521e-04
          2.7707498521e-04 2.7707498521e-04 2.7707498521e-04 2.7707498521e-04 2.7707498521e-04
          2.7707498521e-04 2.7707498521e-04 2.7707498521e-04 2.7707498521e-04 2.7707498521e-04
          2.7707498521e-04 2.7707498521e-04 2.7707498521e-04 2.7707498521e-04 2.7707498521e-04
          2.7707498521e-04 2.7707498521e-04 2.7707498521e-04 2.7707498521e-04 2.7707498521e-04
          2.7707498521e-04 2.7707498521e-04 2.7707498521e-04 2.7707498521e-04 2.7707498521e-04
          2.7707498521e-04 2.7707498521e-04 2.7707498521e-04 2.7707498521e-04 2.7707498521e-04
          2.7707498521e-04

P* SUM    =
          0.0000000000e+00 1.3786630705e-02 5.0408419222e-02 1.0333169252e-01 1.6694033146e-01
          2.3660455644e-01 3.0865362287e-01 3.8029131293e-01 4.4948053360e-01 5.1482015848e-01
          5.7542383671e-01 6.3080912828e-01 6.8080055714e-01 7.2544848919e-01 7.6496040821e-01
          7.9964876175e-01 8.2988816500e-01 8.5608434677e-01 8.7865090370e-01 8.9799290895e-01
          9.1449493170e-01 9.2851579189e-01 9.4038349390e-01 9.5039337873e-01 9.5880943537e-01
          9.6586459875e-01 9.7176229954e-01 9.7668039799e-01 9.8077136278e-01 9.8416715860e-01
          9.8697978258e-01 9.8930495977e-01 9.9122339487e-01 9.9280363321e-01 9.9410295486e-01
          9.9516952038e-01 9.9604386091e-01 9.9675917625e-01 9.9734371901e-01 9.9782061577e-01
          9.9820899963e-01 9.9852484465e-01 9.9878132343e-01 9.9898866681e-01 9.9915689230e-01
          9.9929237366e-01 9.9940139055e-01 9.9948877096e-01 9.9955862761e-01 9.9961411953e-01
          9.9965798855e-01

Q+P* SUM  =
          0.0000000000e+00 1.3954722323e-02 5.0682265311e-02 1.0360575467e-01 1.6721688211e-01
          2.3688152432e-01 3.0893066525e-01 3.8056838512e-01 4.4975760579e-01 5.1509726048e-01
          5.7570093870e-01 6.3108623028e-01 6.8107765913e-01 7.2572559118e-01 7.6523751020e-01
          7.9992586374e-01 8.3016526699e-01 8.5636144876e-01 8.7892800570e-01 8.9827001095e-01
          9.1477203369e-01 9.2879289389e-01 9.4066059589e-01 9.5067048073e-01 9.5908653736e-01
          9.6614170074e-01 9.7203940153e-01 9.7695749998e-01 9.8104846478e-01 9.8444426060e-01
          9.8725688457e-01 9.8958206177e-01 9.9150049686e-01 9.9308073521e-01 9.9438005686e-01
          9.9544662237e-01 9.9632096291e-01 9.9703627825e-01 9.9762082100e-01 9.9809771776e-01
          9.9848610163e-01 9.9880194664e-01 9.9905842543e-01 9.9926596880e-01 9.9943399429e-01
          9.9956947565e-01 9.9967849255e-01 9.9976587296e-01 9.9983572960e-01 9.9989122152e-01
          9.9993509054e-01
```

NUMBER OF FAILED STAGES	UNRELIABILITY AT 80000.0000 HRS	PERFECT COVERAGE UNRELIABILITY AT80000.0000 HRS
0	1.4854960318e-04	0.0000000000e+00
1	1.2852532382e-04	0.0000000000e+00
2	1.4240311292e-16	2.8679277748e-02
3	X	2.8982058167e-01
4	X	6.8115818501e-01

TOTAL SYSTEM UNRELIABILITY AT80000.0000 HRS = 9.9993509054e-01

Example Problem 5: FT = 80,000 hr. - Equal Steps
RELSTP = 1600.0 hr.
NSTEPS = 50

```
 CCCCC         A         RRRR      EEEEE      IIIIIIIIIIIIIII
C             A A        R   R     E            I     I     I
C            A   A       RRRR      EEE          I     I     I
C           A AAA A      R   R     E            I     I     I
 CCCCC     A       A     R    R    EEEEE      IIIIIIIIIIIIIII
```

*** CARE III Workshop (with fault types switched) Example Problem 5 ***

SUBRUN 1

STAGES 1 - 6

CONFIGURATION:

STAGE	N	M
1	3	2
2	3	2
3	4	2
4	1	1
5	1	1
6	3	2

FAULT INFORMATION:

ICAT	RLM	OMG	JTYP
1	1.500e-05	1.00e+00	4
1	1.900e-05	1.00e+00	4
1	7.200e-03	1.00e+00	3
2	3.300e-04	1.00e+00	1
3	4.800e-04	1.00e+00	2
1	1.700e-10	1.00e+00	4
1	2.300e-09	1.00e+00	4
1	3.700e-05	1.00e+00	4

FAULT VECTORS:

TIMES AT WHICH THE FOLLOWING FUNCTION VALUES CORRESPOND (IN HRS):

0.0000000000e+00 1.1920928955e-03 2.3841857910e-03 4.7683715820e-03 7.1525573730e-03
1.1920928955e-02 1.6689300537e-02 2.6226043701e-02 3.5762786865e-02 5.4836273193e-02
7.3909759521e-02 1.1205673218e-01 1.5020370483e-01 2.2649765015e-01 3.0279159546e-01
4.5537948608e-01 6.0796737671e-01 9.1314315796e-01 1.2183189392e+00 1.8286705017e+00
2.4390220642e+00 3.6597251892e+00 4.8804283142e+00 7.3218345642e+00 9.7632408142e+00
1.4646053314e+01 1.9528865814e+01 2.9294490814e+01 3.9060115814e+01 5.8591365814e+01
7.8122619629e+01 1.1718511963e+02 1.5624761963e+02 2.3437261963e+02 3.1249761963e+02
4.6874761963e+02 6.2499761963e+02 9.3749761963e+02 1.2499976807e+03 1.8749976807e+03
2.4999978027e+03 3.7499978027e+03 4.9999980469e+03 7.4999980469e+03 9.9999980469e+03
1.4999998047e+04 1.9999998047e+04 2.9999998047e+04 4.0000000000e+04 6.0000000000e+04
8.0000000000e+04

.

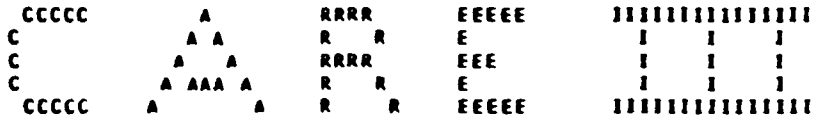
```
.
.
SUMMARY INFORMATION:

Q SUM =
0.0000000000e+00 7.2471034729e-10 2.7339039921e-09 9.1005043501e-09 1.6784373003e-08
3.3429358837e-08 5.0584468170e-08 8.5190258403e-08 1.1985576975e-07 1.8919764955e-07
2.5854382102e-07 3.9722650058e-07 5.3589553772e-07 8.1319120682e-07 1.0904312830e-06
1.6447451117e-06 2.1988369099e-06 3.0063531646e-06 4.4129842536e-06 6.6235907070e-06
8.8306469479e-06 1.3234161997e-05 1.7623588064e-05 2.6360374250e-05 3.5041393858e-05
5.2237686759e-05 6.9215595431e-05 1.0252831999e-04 1.3500345813e-04 1.9753280503e-04
2.5698074023e-04 3.6728935083e-04 4.6715099597e-04 6.3975137891e-04 7.8209425556e-04
9.9799840245e-04 1.1484518182e-03 1.3315024553e-03 1.4285265934e-03 1.5145826619e-03
1.5461129369e-03 1.5663161175e-03 1.5712700551e-03 1.5738148941e-03 1.5741386451e-03
1.5743492404e-03 1.5743690310e-03 1.5743748518e-03 1.5743749682e-03 1.5743749682e-03
1.5743749682e-03

P* SUM =
0.0000000000e+00 8.3346661346e-15 3.3338661150e-14 1.3335463105e-13 3.0004788259e-13
8.3346621705e-13 1.6335933257e-12 4.0339732423e-12 7.5011854367e-12 1.7636104382e-11
3.2038337261e-11 7.3644687570e-11 1.3232014096e-10 3.0087790592e-10 5.3771093134e-10
1.2161992524e-09 2.1677786233e-09 4.8901838134e-09 8.7048759312e-09 1.9610908453e-08
3.4885459854e-08 7.8538469950e-08 1.3966059953e-07 3.1429883052e-07 5.5877364957e-07
1.2571265415e-06 2.2345075195e-06 5.0255380302e-06 8.9298937382e-06 2.0073561245e-05
3.5649805795e-05 8.0051700934e-05 1.4202721650e-04 3.1826479244e-04 5.6350952946e-04
1.2575592846e-03 2.2173072211e-03 4.9065775238e-03 8.5769798607e-03 1.8648095429e-02
3.2014854252e-02 6.7066125572e-02 1.1080279201e-01 2.1442399919e-01 3.2669317722e-01
5.3813493252e-01 7.0378243923e-01 8.9343804121e-01 9.6586459875e-01 9.9706619978e-01
9.9965798855e-01

Q+P* SUM =
0.0000000000e+00 7.2471867396e-10 2.7339372988e-09 9.1006375769e-09 1.6784673207e-08
3.3430193724e-08 5.0586102418e-08 8.5194294286e-08 1.1986327308e-07 1.8921528522e-07
2.5857585229e-07 3.9730014123e-07 5.3602786920e-07 8.1349207903e-07 1.0909690218e-06
1.6459613335e-06 2.2010046905e-06 3.3112432902e-06 4.4216890274e-06 6.6432016865e-06
8.8655324362e-06 1.3312700503e-05 1.7763248252e-05 2.6674673791e-05 3.5600169213e-05
5.3494812164e-05 7.1450100222e-05 1.0755386029e-04 1.4393335732e-04 2.1760637173e-04
2.9263054603e-04 4.4734106632e-04 6.0917821247e-04 9.5801620046e-04 1.3456038432e-03
2.2555578034e-03 3.3657590393e-03 6.2380800955e-03 1.0005506687e-02 2.0162677392e-02
3.3560968935e-02 6.8632438779e-02 1.1237405986e-01 2.1599781513e-01 3.2826730609e-01
5.3970927000e-01 7.0535683632e-01 8.9501243830e-01 9.6743899584e-01 9.9864059687e-01
1.0012323856e+00

=

NUMBER OF FAILED STAGES | UNRELIABILITY AT 80000.0000 HRS | PERFECT COVERAGE UNRELIABILITY AT80000.0000 HRS

0 | 1.4295593137e-03 | 0.0000000000e+00
1 | 1.4481593098e-04 | 0.0000000000e+00
2 | X | 2.8679277748e-02
3 | X | 2.8982058167e-01
4 | X | 6.8115818501e-01

TOTAL SYSTEM UNRELIABILITY AT80000.0000 HRS = 1.0012323856e+00
```

Example Problem 5: FT = 80,000 hr. - Unequal Steps
CVGSTP = 1.1921e-3 hr.
NPERST = 2
NDUP = 24
NSTEPS = 50

This Page Intentionally Left Blank

APPENDIX C

FTMP Output Listing



FTMP ARCHITECTURE - 15 PROCESSORS, 9 MEMORY MODULES, 5 BUSES
WITH CRITICAL FAULT PAIRS AND INTERNALLY REDUNDANT MEMORY MODULES.
NOVEMBER, 1984 - With RLM = ORIGINAL LAMBDA and RLMSUB = 0.0.
THIS SHOULD YIELD THE EXACT SAME RESULTS AS THE TEST CASE NOT USING
INTERNALLY REDUNDANT MODULES.

SUBRUN 1

STAGES 1 - 3

CONFIGURATION:

STAGE	N	M	NSUB	MSUB	ACTIVE SPARES
1	15	11	0	0	
2	9	5	44	40	F
3	5	3	0	0	

FAULT INFORMATION:

ICAT	RLM	OMG	JTYP
1	1.000e-04	1.00e+00	1
2	1.800e-05	1.00e+00	1
1 P	1.000e-04	1.00e+00	3
1 T	0.000e+00	1.00e+00	3
2 P	1.800e-05	1.00e+00	3
2 T	0.000e+00	1.00e+00	3
1	1.000e-06	1.00e+00	2

FAULT VECTORS:

TIMES AT WHICH THE FOLLOWING FUNCTION VALUES CORRESPOND (IN MINS):

0.0000000000e+00 1.5873016417e-01 3.1746032834e-01 4.7619050741e-01 6.3492065668e-01
7.9365080595e-01 9.5238095522e-01 1.1111111641e+00 1.2698413134e+00 1.4285714626e+00
1.5873016119e+00 1.9047619104e+00 2.2222223282e+00 2.5396826267e+00 2.8571429253e+00
3.1746032238e+00 3.4920635223e+00 3.8095238209e+00 4.1269841194e+00 4.4444446564e+00
4.7619051933e+00 5.3968257904e+00 6.0317463875e+00 6.6666669846e+00 7.3015875816e+00
7.9365081787e+00 8.5714292526e+00 9.2063503265e+00 9.8412714005e+00 1.0476192474e+01
1.1111113548e+01 1.2380954742e+01 1.3650795937e+01 1.4920637131e+01 1.6190479279e+01
1.7460321426e+01 1.8730163574e+01 2.0000005722e+01 2.1269847870e+01 2.2539690018e+01
2.3809532166e+01 2.6349214554e+01 2.8888896942e+01 3.1428579330e+01 3.3968261719e+01
3.6507946014e+01 3.9047630310e+01 4.1587314606e+01 4.4126998901e+01 4.6666683197e+01
4.9206367493e+01 5.4285732269e+01 5.9365097046e+01 6.4444465637e+01 6.9523834229e+01
7.4603202820e+01 7.9682571411e+01 8.4761940002e+01 8.9841308594e+01 9.4920677185e+01
1.0000004578e+02

SUMMARY INFORMATION:

Q SUM =

0.0000000000e+00 1.3021668162e-12 4.9515834141e-12 1.0631093228e-11 1.8097587665e-11
2.7163417349e-11 3.7684279308e-11 4.9549600534e-11 6.2675718782e-11 7.7005929411e-11
9.2436627797e-11 1.2483350464e-10 1.5755381721e-10 1.9075230284e-10 2.2427118496e-10
2.5802338044e-10 2.9193900075e-10 3.2596983446e-10 3.5999328518e-10 3.9398581619e-10
4.2801825972e-10 4.9616138975e-10 5.6438903551e-10 6.3268867923e-10 7.0106043193e-10
7.6950096295e-10 8.3801082740e-10 9.0658880403e-10 9.7523455977e-10 1.0439470399e-09
1.1127254673e-09 1.2504802749e-09 1.3884950967e-09 1.5267643816e-09 1.6652839108e-09
1.8040499095e-09 1.9430581588e-09 2.0823043290e-09 2.2217838680e-09 2.3614936673e-09
2.5014288418e-09 2.7819635484e-09 3.0633540149e-09 3.3455691550e-09 3.6285809912e-09
3.9123606577e-09 4.1968735154e-09 4.4820933631e-09 4.7679975523e-09 5.0545612140e-09
5.3417590351e-09 5.9179452450e-09 6.4963789903e-09 7.0769141658e-09 7.6593975606e-09
8.2437034976e-09 8.8296197021e-09 9.4170173881e-09 1.0005866358e-08 1.0596054700e-08
1.1187474058e-08

P* SUM =

0.0000000000e+00 1.8515966906e-25 1.4814999099e-24 5.0013150302e-24 1.1859123482e-23
2.3172784358e-23 4.0064601390e-23 6.3662476068e-23 9.5100943413e-23 1.3552234028e-22
1.8607793658e-22 3.2224778652e-22 5.1304115494e-22 7.6810187273e-22 1.0973231392e-21
1.5108838600e-21 2.0192819690e-21 2.6333726970e-21 3.3644036608e-21 4.2240478812e-21
5.2244436683e-21 7.6985554207e-21 1.0894455767e-20 1.4929527613e-20 1.9931991225e-20
2.6042028674e-20 3.3412920406e-20 4.2212199575e-20 5.2622739335e-20 6.4843974225e-20
7.9092993920e-20 1.1463777202e-19 1.6138994269e-19 2.2183800126e-19 2.9886200968e-19
3.9577035434e-19 5.1633559923e-19 6.6483147639e-19 8.4607010902e-19 1.0654356157e-18
1.3289223159e-18 2.0155120029e-18 2.9673845093e-18 4.2581723514e-18 5.9747656827e-18
8.2184678653e-18 1.1106157038e-17 1.4771455544e-17 1.9365889635e-17 2.5060022301e-17
3.2044689071e-17 5.0756894265e-17 7.7477488263e-17 1.1457941732e-16 1.6487042362e-16
2.3163000033e-16 3.1864717731e-16 4.3025631479e-16 5.7137523311e-16 7.4754209840e-16
9.6495062730e-16

Q+P* SUM =

0.0000000000e+00 1.3021668162e-12 4.9515834141e-12 1.0631093228e-11 1.8097587665e-11
2.7163417349e-11 3.7684279308e-11 4.9549600534e-11 6.2675718782e-11 7.7005929411e-11
9.2436627797e-11 1.2483350464e-10 1.5755381721e-10 1.9075230284e-10 2.2427118496e-10
2.5802338044e-10 2.9193900075e-10 3.2596983446e-10 3.5999328518e-10 3.9398581619e-10
4.2801825972e-10 4.9616138975e-10 5.6438903551e-10 6.3268867923e-10 7.0106043193e-10
7.6950096295e-10 8.3801082740e-10 9.0658880403e-10 9.7523455977e-10 1.0439470399e-09
1.1127254673e-09 1.2504802749e-09 1.3884950967e-09 1.5267643816e-09 1.6652839108e-09
1.8040499095e-09 1.9430581588e-09 2.0823043290e-09 2.2217838680e-09 2.3614936673e-09
2.5014288418e-09 2.7819635484e-09 3.0633540149e-09 3.3455691550e-09 3.6285809912e-09
3.9123606577e-09 4.1968735154e-09 4.4820933631e-09 4.7679975523e-09 5.0545612140e-09
5.3417590351e-09 5.9179452450e-09 6.4963789903e-09 7.0769141658e-09 7.6593975606e-09
8.2437034976e-09 8.8296197021e-09 9.4170173881e-09 1.0005867246e-08 1.0596055589e-08
1.1187474946e-08

NUMBER OF FAILED STAGES	UNRELIABILITY AT 100.0000 MINS	PERFECT COVERAGE UNRELIABILITY AT 100.0000 MINS
0	1.1187474058e-08	0.0000000000e+00
1	X	9.6495062730e-16

TOTAL SYSTEM UNRELIABILITY AT 100.0000 MINS= 1.1187474946e-08

FTMP Modified Test Case (with internally redundant memory modules)

FT = 100 min. - Unequal Steps
CVGSTP = 1.5873e-1 min.
NPERST = 10
NDUB = 5
NSTEPS = 60

End of Document